PERFORCE 2008.1
P4 ユーザーズ・ガイド

July 2008

目　次

使用例一覧

はじめに

このマニュアルについて

本書では、PERFORCE のコマンドライン・クライアント (p4) の使用方法を説明します。ソフトウェア構成管理 (SCM) ツールを初めてお使いになる場合、PERFORCE の基本概念をご存知でない場合、または PERFORCE を一度も使用したことがない場合は、本書をお読みになる前に『*PERFORCE 概要*』をご覧ください。本書では SCM に関して十分な基礎知識があることを前提としています。

コマンドラインと GUI

PERFORCE には、PERFORCE コマンドライン・クライアント、P4V などの GUI ツール、プラグインなど、ファイル管理を行うためのクライアント・アプリケーションが数多く用意されています。PERFORCE コマンドライン・クライアントにより、PERFORCE GUI ではサポートされていない、システム管理作業のスクリプト化や実行を行うことができます。

PERFORCE 入門

PERFORCE を初めて操作する場合は、次の手順に従ってください。

1. 『*PERFORCE 概要*』を読み、基礎を習得します。

   少なくとも、*チェンジリスト*、*ディポ*、*クライアント・ワークスペース*、*同期*、*サブミット*の概念について習得してください。簡潔な用語定義が、本書の付録にある用語集に記述されています。

2. PERFORCE 管理者に PERFORCE サーバのホストおよびポートの情報を問い合わせてください。

   実稼動のディポを破壊することなく PERFORCE の試験を行いたい場合は、試験用の別のサーバを起動するよう PERFORCE 管理者に依頼してください。PERFORCE サーバのインストールに関する詳細は、『*PERFORCE システム管理者ガイド*』を参照してください。

3. システム管理者が既にマシンの構成を行っていない場合、本書を参照して PERFORCE コマンドライン・クライアントのインストールおよびクライアント・ワークスペースの構成を行ってください。詳しくは、第 2 章の「P4 の構成」をご覧ください。

4. 以下の作業の実行方法を習得してください。

- *同期* (選択したファイルをリポジトリから自分のコンピュータに転送します)
- *サブミット* (変更済みのファイルを自分のワークスペースからリポジトリに転送します)
- *元に戻す* (変更を破棄します)

   詳しくは、第 4 章の「ファイルとチェンジリストの管理」をご覧ください。

5. 自分のクライアント・ビューの限定方法を習得してください。詳しくは、19 ページの「クライアント・ビューの限定」をご覧ください。

これらについて基本的な技術があれば、日常業務の多くをこなすことができます。その他のコードラインの保守（ブランチ操作とラベル付け）やワークフロー（ジョブ）にかかわる作業を実施する頻度はそれほど多くありません。本書では p4 コマンドを使用したこれらの作業の実施方法についても説明しています。

PERFORCE ドキュメント

本書、*『PERFORCE コマンド・リファレンス』*、および p4 help コマンドが PERFORCE コマンドライン・クライアントに関する主要なドキュメントです。本書では最新版のリリースについて説明しています。以前のリリースのドキュメントについては PERFORCE 社の Web サイトを参照してください。

他の PERFORCE クライアント・プログラムに関するドキュメントは、PERFORCE 社のウェブサイト
http://www.perforce.com の documentation のページから入手できます。

記述される詳細情報	記述ドキュメント
PERFORCE の基礎	*『PERFORCE 概要』*
PERFORCE サーバ、プロキシサーバ、セキュリティ設定のインストールおよび管理	*『PERFORCE システム管理者ガイド』*
p4 コマンドライン・フラグおよびオプション（リファレンス）	*『PERFORCE コマンド・リファレンス』*、p4 help
P4V ープラットフォーム互換 PERFORCE ビジュアル・クライアント	*『P4V 入門』*、P4V オンライン・ヘルプ
P4Web ーブラウザベースの PERFORCE クライアント・アプリケーション	*『P4Web の使用方法』*、P4Web オンラインヘルプ
P4Win ー PERFORCE Windows GUI	*『P4Win 入門』*、P4Win オンラインヘルプ
PERFORCE プラグイン	IDE : *『IDE プラグイン・ユーザーズ・ガイド』* その他 : PERFORCE メニューからのオンラインヘルプ
PERFORCE C/C++ API を利用した PERFORCE クライアント・アプリケーションの開発	*『C/C++ API ユーザ・ガイド』*

注 日本語版 PERFORCE は、P4Web, C/C++ API をサポートしておりませんのでご注意ください。

ご意見・ご感想をお待ちしています

当社では、本書に関するユーザのみなさんからのご意見をお待ちしています。とりわけ、初めて PERFORCE をお使いになったユーザのみなさんのご意見は、ぜひお聞かせ願いたいと思います。本書を読んで十分な情報は得られましたか？ ご感想をお聞かせください。ご意見・ご感想の宛先は下記のとおりです。
manual@perforce.com

第 1 章

P4 のインストール

この章では、PERFORCE コマンドライン・クライアント (p4) をクライアント・マシンにインストールする方法を説明します。PERFORCE サーバのインストールに関して詳しくは、*『PERFORCE システム管理者ガイド』* を参照してください。

UNIX および OS X への P4 のインストール

PERFORCE コマンドライン・クライアントを UNIX または Macintosh OS X マシンにインストールする手順を以下に示します。

1. p4 実行可能ファイルを以下の PERFORCE 社の Web サイトからダウンロードします。

   `http://www.perforce.com/perforce/loadprog.html`

   PERFORCE クライアント・プログラムは通常 `/usr/local/bin` にダウンロードされます。

2. p4 ファイルを実行可能にします (`chmod +x p4`)。
3. サーバ・ポートの設定、クライアント・ワークスペース名、およびユーザ名を構成します。`P4PORT`、`P4CLIENT`、`P4USER` の各環境変数により、これらの設定を行うことができます。(詳しくは、第 2 章の「P4 の構成」を参照してください)。

Windows への P4 のインストール

PERFORCE コマンドライン・クライアント (`p4.exe`) を Windows にインストールするには、以下の PERFORCE 社の Web サイトにある Downloads ページから、PERFORCE Windows インストーラ (`perforce.exe`) をダウンロードして実行します。

`http://www.perforce.com/perforce/loadprog.html`

PERFORCE インストーラにより、PERFORCE コマンドライン・クライアントおよびその他の PERFORCE Windows コンポーネントのインストールとアンインストールを行うことができます。

インストールの確認

PERFORCE コマンドライン・クライアントが正常にインストールされたかどうか確認するには、コマンドラインで p4 info と入力して ENTER キーを押します。サーバが特定のホストおよびポートで稼動していれば、次のようなメッセージが表示されます。

```
User name: ona
Client name: ona-agave
Client host: agave
Client root: /home/ona/p4-ona
Current directory: /home/ona/p4-ona
Client address: 10.0.0.196:2345
Server address: ida:1818
Server root: /usr/depot/p4d
Server date: 2005/06/13 08:46:34 -0700 PDT
Server version: P4D/FREEBSD4/2005.1/79540 (2005/05/10)
Server license: PERFORCE Software 200 users (expires 2007/01/31)
```

構成の設定が正しくない場合、次のようなメッセージが表示されます。

```
PERFORCE client error:
        Connect to server failed; check $P4PORT.
        TCP connect to <hostname> failed.
        <hostname>: host unknown.
```

第 2 章

P4 の構成

この章では、接続の設定の構成方法について説明します。

構成の概要

PERFORCE ではクライアント / サーバのアーキテクチャが使用されています。ユーザは*ディポ*と呼ばれるサーバのリポジトリからファイルを*同期*させ、ユーザのクライアント・マシン上の*クライアント・ワークスペース*においてそれらのファイルを編集します。この章では、システム管理者が既に PERFORCE サーバを起動していることを想定しています。PERFORCE サーバのインストールに関して詳しくは、*『PERFORCE システム管理者ガイド』* を参照してください。

クライアント・ワークスペースを設定してサーバとのやり取りを可能にするには、次の手順に従ってください。

ステップ #	詳細の記述箇所
1. 使用するサーバのホストおよびポートを設定（PERFORCE サーバの起動場所を指定）します。	15 ページの「Perforce の設定」
2. クライアント・ワークスペースを定義します（少なくとも、名前を指定し、ディポ・ファイルのローカル・コピーを格納したいワークスペースのルートを指定してください）。	17 ページの「クライアント・ワークスペースの定義」
3. 接続を確認します。	19 ページの「接続の確認」

ワークスペースを構成したら、ディポに格納されているファイルと同期させてワークスペースに取り込むことができます。詳しくは、42 ページの「ファイルの同期（取得）」および *『PERFORCE コマンド・リファレンス』* の `p4 sync` コマンドに関する説明を参照してください。

クライアント・マシンの構成を始める前に、PERFORCE 管理者にサーバ・ホストおよびポートの設定について問い合わせてください。まは、クライアント・マシンに既にワークスペースが構成されているか確認してください。

クライアント・ワークスペースとは

PERFORCE *クライアント・ワークスペース*とは、PERFORCE によって管理されるファイル・リビジョンの操作を行うクライアント・マシン上にある、ディレクトリの集合です。各ワークスペースには、PERFORCE サーバに対してクライアント・ワークスペースを識別するための固有の名前が付いています。`P4CLIENT` 環境変数の設定によりワークスペース名を指定しない場合、デフォルトのワークスペース名はクライアント・マシンの名前になります。効果的なワークスペース名を指定するため、`P4CLIENT` 環境変数を設定してください。1 つのクライアント・マシンに複数のワークスペースを含めることができます。

PERFORCE クライアント・ワークスペースにあるすべてのファイルは、*クライアント・ルート*と呼ばれるルート・ディレクトリを共有します。クライアント・ルートはワークスペースの最上位ディレクトリであり、その配下に管理対象のソース・ファイルが格納されます。

1 つのマシン上に複数のワークスペースを構成する場合、不用意にファイルが上書きされることのないよう、ワークスペースの場所を分離してください。クライアント・ルートが別々のフォルダに配置されるようにし、クライアント・ビューをディポ・ファイルにマッピングするとき、クライアント・マシン上の位置が重ならないようにしてください。

ワークスペースを構成したら、ファイルをディポと同期して変更をサブミットすることができます。これらの操作について詳しくは、第 4 章の「ファイルとチェンジリストの管理」を参照してください。

PERFORCE によるワークスペースの管理方法

PERFORCE はクライアント・ワークスペース内のファイルを以下のように管理します。

- ユーザによる変更に応じて、ワークスペース内のファイルの作成、更新、削除が行われます。
- 書き込み許可はユーザがファイルを編集する際に有効にされ、変更をサブミットする際に無効にされます。

ユーザのクライアント・ワークスペースの状態は、PERFORCE サーバにより追跡され、管理されます。サーバにより実行されるファイル管理との衝突を避けるため、ファイルの読み取り専用の許可設定を手動で変更しないでください。クライアント・ワークスペースの状態が、サーバに記録されているそのワークスペースの状態と一致しているかを検証できます。詳しくは、PERFORCE Web サイトにある Technote2 を参照してください。

```
http://www.perforce.com/perforce/technotes/note002.html
```

PERFORCE により制御されるよう指定していないワークスペース内のファイルは、PERFORCE には無視されます。例えば、コンパイル済みオブジェクト、ライブラリ、実行可能ファイル、およびソフトウェアの開発中に作成されたディポに追加されていない開発者用一時ファイルなどは、PERFORCE コマンドの影響を受けません。

クライアント・ワークスペースを定義したら、ワークスペースの定義を微調整することができます。おそらく最も重要なことは、ディポ内のユーザが見ることのできる部分を制限して、不用意にディポ全体を同期させないようにすることです。詳しくは、19 ページの「クライアント・ビューの限定」を参照してください。

PERFORCE の設定

本書では、環境変数を使用した PERFORCE の設定方法 (「set P4CLIENT」など) を説明していますが、サーバ・ポート、ユーザ、ワークスペース名などの PERFORCE の設定は以下の方法で指定することができます。それらについて優先順位の高い順に示します。

1. コマンドラインでオプションを使用する
2. P4CONFIG が設定されている場合、構成ファイルで指定する
3. ユーザ環境変数（UNIX または Windows において）を使用する
4. システム環境変数を使用する（Windows ではシステム全体の環境変数はユーザ環境変数と同じではないことがあります）
5. Windows において、PERFORCE ユーザ・レジストリを使用する（p4 set コマンドを発行して設定する）
6. Windows において、PERFORCE システム・レジストリを使用する（p4 set -s コマンドを発行して設定する）

PERFORCE サーバに接続するようクライアント・マシンを構成するには、サーバが稼働しているホストの名前およびサーバが接続を待機するポートを指定します。デフォルトのサーバ・ホストは PERFORCE、デフォルトのサーバ・ポートは 1666 です。サーバがクライアント・マシン上で起動している場合、ホスト名を localhost に指定してください。サーバがポート 1666 で起動している場合、ポートの指定を省略することができます。

以降のセクションで説明するとおり、これらの設定を指定することができます。非接続状態 (PERFORCE サーバに接続しない）での作業に関する詳細は、51 ページの「非接続で作業する」を参照してください。

コマンドラインの使用

サーバ設定をコマンドラインで指定するには、-p オプションを使用してください。例えばこのようにします。

```
p4 -p localhost:1776 sync //depot/dev/main/jam/Jambase
```

コマンドラインで指定されたサーバ設定は、構成ファイル、環境変数、または Windows レジストリで指定された設定に優先します。コマンドライン・オプションに関する詳細は、*『PERFORCE コマンド・リファレンス』*のグローバル・オプションの説明を参照してください。

構成ファイルの使用

*構成ファイル*は、そのファイルのあるディレクトリおよびその配下のディレクトリにあるファイルに影響する PERFORCE の設定が記述されているテキスト・ファイルです。構成ファイルは、1 つのマシン上に複数のクライアント・ワークスペースがある場合に有用です。構成ファイルで設定を指定すれば、異なるワークスペースで作業するたびにシステム設定を変更するという面倒な作業は必要ありません。

構成ファイルを使用するには、ファイル名を指定（.p4config など）して、P4CONFIG 環境変数を定義します。コマンドの発行時、PERFORCE は現在の作業ディレクトリおよびその親ディレクトリで指定されたファイルを探し、(設定がコマンドライン・オプションにより上書きされない限り）そのファイルに含まれる設定を使用します。

ファイル内では、各設定につき 1 行ずつ以下の形式で指定する必要があります。

設定 = 値

構成ファイルで指定できる設定項目を以下に示します。

設定	説明
P4CHARSET	Unicode ファイルの翻訳に使用される文字セット
P4COMMANDCHARSET	P4CHARSET が UTF-16 文字セットまたは UTF-32 文字セットに設定されている場合にコマンドライン・クライアントで使用される UTF-16 または UTF-32 以外の文字セット
P4CLIENT	現在のクライアント・ワークスペース名
P4DIFF	p4 resolve および p4 diff により使用される比較プログラムの名前および場所
P4EDITOR	フォームを使用する PERFORCE コマンドにより起動されるエディタ
P4HOST	クライアント・ワークステーションのホスト名。現在のクライアント・ワークスペースの Host: フィールドが p4 client フォームに設定されている場合にのみ有用です。
P4LANGUAGE	この環境変数はシステムの統合用に予約されています。
P4MERGE	p4 resolve のマージ・オプションで使用するサード・パーティのマージ・プログラムの名前および場所
P4PASSWD	PERFORCE クライアント・コマンドに対し、現在の PERFORCE ユーザのパスワードを指定します。
P4PORT	接続先の PERFORCE サーバまたはプロキシのホストおよびポート番号
P4USER	現在の PERFORCE ユーザ名

これらの設定に関して詳しくは、*『PERFORCE コマンド・リファレンス』* を参照してください。

例 : *構成ファイルを使用して 2 つのワークスペースを切り替える*

オナは 1 つのマシン上の 2 つのワークスペースを切り替えます。1 番目のワークスペースは ona-ash *です。そのクライアント・ルートは* /tmp/user/ona *であり、PERFORCE サーバに* ida:1818 *で接続します。2 番目のワークスペースは* ona-agave *で、そのクライアント・ルートは* /home/ona/p4-ona *であり、PERFORCE サーバに* warhol:1666 *で接続します。*

オナは P4CONFIG *環境変数を* .p4settings *に設定します。彼女は* /tmp/user/ona *に以下のテキストを含む* .p4settings *というファイルを作成します。*

```
P4PORT=ida:1818
P4CLIENT=ona-ash
```

彼女は /home/ona/p4-ona *に 2 番目の* .p4settings *ファイルを作成します。そのファイルには次のテキストが含まれます。*

```
P4PORT=warhol:1666
P4CLIENT=ona-agave
```

彼女が /tmp/user/ona *配下のファイルに対して行う作業は、*ida:1818 *にある PERFORCE サーバにより管理されます。*/home/ona/p4-ona *配下のファイルに対して行う作業は、*warhol:1666 *にある PERFORCE サーバにより管理されます。*

環境変数の使用

環境変数を使用してサーバ接続設定を構成するには、P4PORT を以下の例のように *host:port* の形式で設定してください。

サーバが稼動している場所	接続を待機しているポート	P4PORT の設定
ユーザのコンピュータ	1666	1666
perforce	1666	1666

サーバが稼動している場所	接続を待機しているポート	P4PORT の設定
houston	3435	houston:3435
deneb.com	1818	deneb.com:1818

Windows レジストリの使用

Windows マシンでは、p4 set コマンドを発行することにより、レジストリに接続設定を保存することができます。例えばこのようにします。

```
p4 set P4PORT=tea:1667
```

PERFORCE の設定をレジストリに構成する方法には以下の２つがあります。

- p4 set *setting=value* ー 現在の Windows ログイン用。
- p4 set -s *setting=value* ー ローカル・マシン上のすべてのユーザ用。ローカル・ユーザ用のレジストリ設定を上書きします。設定には PERFORCE 管理者権限が必要です。

現在の設定を確認するには、p4 set コマンドを入力してください。p4 set コマンドに関して詳しくは、*『PERFORCE コマンド・リファレンス』* を参照してください。

クライアント・ワークスペースの定義

クライアント・ワークスペースを定義するには、

1. P4CLIENT を設定してワークスペース名を指定します。例えば UNIX システムの場合、以下のようにします。
   ```
   $ P4CLIENT=bruno_ws ; export P4CLIENT
   ```
2. p4 client コマンドを発行します。

   PERFORCE はクライアント仕様フォームをユーザのテキスト・エディタに表示します。(PERFORCE のフォームに関して詳しくは、39 ページの「Perforce フォームの使用」を参照してください。)
3. （最小限必要な）設定を指定し、仕様を保存します。

クライアント仕様を作成するとき、ファイルは一切同期されません。ファイルをディポからワークスペースに同期する方法を知るには、42 ページの「ファイルの同期（取得)」を参照してください。マシン上でのファイルの移動に関して詳しくは、25 ページの「ワークスペースの場所を変更する」を参照してください。

クライアント・ワークスペースの構成において最小限指定しなければならない設定は以下の通りです。

- **ワークスペース名**

  ワークスペース名はクライアント・マシンのホスト名をデフォルトとしますが、クライアント・マシンには複数のワークスペースを含めることができます。効果的にワークスペースを指定するには、P4CLIENT を設定してください。

- **クライアント・ルート**

  クライアント・ルートはクライアント・ワークスペースの最上位のディレクトリであり、PERFORCE はそこにディポ・ファイルの作業用コピーを保存します。クライアント・ルートを必ず設定してください。そうしないと、ファイルを不用意に自分のクライアント・マシンのルート・ディレクトリに同期させてしまう可能性があります。

*クライアント・ビュー*により、ディポ内のどのファイルがクライアント・ワークスペースにマッピングされるかが決定され、サーバが個々のディポとワークスペース・ファイルの間に 1 対 1 のマッピングを構成することができます。ディポ内での名前や場所とは異なる名前およびワークスペース内の場所にファイルをマッピングすることができますが、ワークスペースまたはディポ内の複数の場所にファイルをマッピングすることはできません。デフォルトでは、ディポ全体がユーザのワークスペースにマッピングされます。クライアント・ビューを定義し、自分に関係するファイルおよびディレクトリのみをマッピングすることにより、ディポ全体を不用意に自分のワークスペースに同期させないようにすることができます。詳しくは、19 ページの「クライアント・ビューの限定」を参照してください。

例 : *クライアント・ビューの設定*

*ブルーノは、*p4 client *コマンドを発行して、以下のデフォルトのクライアント・ビュー定義を含むフォームを参照します。*

```
Client:     bruno_ws
Update:     2004/11/29 09:46:53
Access:     2005/03/02 10:28:40
Owner:     bruno
Root:     c:\bruno_ws
Options: noallwrite noclobber nocompress unlocked nomodtime normdir
SubmitOptions: submitunchanged
LineEnd: local
View:
        //depot/...      //bruno_ws/...
```

彼は、ビューを変更してディポの開発部分のみをマッピングするようにします。

```
View:
        //depot/dev/...      //bruno_ws/dev/...
```

彼はさらにビューを変更して、ファイルを複数のディポから自分のワークスペースにマッピングします。

```
View:
        //depot/dev/...      //bruno_ws/depot/dev/...
        //testing/...        //bruno_ws/testing/...
        //archive/...        //bruno_ws/archive/...
```

接続の確認

接続状態を確認するには、p4 info コマンドを発行します。P4PORT が正しく設定されていれば、次のような情報が表示されます。

```
User name: bruno
Client name: bruno_ws
Client host: workstation_12
Client root: c:\bruno_ws
Current directory: c:\bruno_ws
Client address: 127.0.0.1:28
Server address: localhost:1667
Server root: /usr/depot/p4d
Server date: 2008/06/28 15:03:05 -0700 PDT
Server uptime: 752:41:23
Server version: P4D/FREEBSD4/2008.1/156375 (2008/05/25)
Proxy version: P4P/SOLARIS26/2008.1/156884 (2008/05/25)
Server license: P4 Admin <p4adm> 20 users (expires 2009/01/01)
```

Server address:フィールドにはp4が接続している PERFORCE サーバ､およびPERFORCEサーバが接続を待機しているホスト名とポート番号が表示されます。P4PORT の設定が正しくない場合、次のようなメッセージを受け取ります。

```
PERFORCE client error:
       Connect to server failed; check $P4PORT.
       TCP connect to perforce:1666 failed.
       perforce: host unknown.
```

エラー・メッセージの 3 行目に表示される値が perforce:1666（上記のとおり）である場合、P4PORT が設定されていません。それ以外の値である場合は、P4PORT の設定が不正です。

クライアント・ビューの限定

デフォルトでは、クライアント・ワークスペースを作成すると、ディポ全体がマッピングされます。このマッピングをディポの一部のみを参照するように限定し、ディポとワークスペースの場所との対応関係を変更することができます。

クライアント・ビューを表示または変更するには、p4 client コマンドを発行します。PERFORCE によりクライアント仕様フォームの [View:] フィールドにマッピングのリストが表示されます。

```
Client: bruno_ws
Owner:  bruno
Description:
         Created by bruno.
Root:  C:\bruno_ws
Options: noallwrite noclobber nocompress unlocked nomodtime normdir
SubmitOptions: submitunchanged
View:
        //depot/...   //bruno_ws/...
```

以降のセクションにおいて、クライアント・ビューの指定に関して詳しく説明します。より詳しい情報は、*『PERFORCE コマンド・リファレンス』*の p4 client コマンドの説明およびビューの説明を参照してください。

マッピングの指定

ビューは複数の *マッピング*で構成され、各マッピングは 2 つの部分から成ります。

- 左側には、ディポ内の 1 つ以上のファイルが次の形式で指定されます。
  //depotname/file_specification
- 右側には、クライアント・ワークスペース内の 1 つ以上のファイルが次の形式で指定されます。
  //clientname/file_specification

クライアント・ビュー・マッピングの左側を *ディポ・サイド*、右側を *クライアント・サイド*と呼びます。

クライアント・ファイルのホスト・マシン上での場所を決定するには、クライアン・ルートをマッピングのクライアント・サイドのクライアント名に置き換えてください。例えば、クライアント・ルートが C:\bruno_ws である場合、//depot/dev/main/jam/Jamfile ファイルはワークスペース内に C:¥bruno_ws¥dev¥main¥jam¥Jamfile として格納されます。

後に記述されるマッピングが、それより前のマッピングより優先されます。以下の例では、2 行目が 1 行目より優先され、//depot/dev/main/docs/manuals/ 内のファイルが 2 階層上にマッピングされます。//depot/dev/main/docs/manuals/ 内のファイルが同期されると、それらのファイルは c:¥bruno_ws¥docs¥ に格納されます。

```
View:
    //depot/dev/...             //bruno_ws/dev/...
    //depot/dev/main/docs/...   //bruno_ws/docs/...
```

クライアント・ビューでのワイルドカードの使用

ファイルの集合をクライアント・ビューにマッピングするには、PERFORCE ワイルドカードを使用します。マッピングのディポ・サイドで使用されるワイルドカードが、クライアント・サイドのワイルドカードと一致する必要があります。クライアント・ビュー・マッピングの指定には、以下のワイルドカードを使用できます。

ワイルドカード	説明
*	スラッシュを除くすべての文字に一致します。単一のディレクトリ内のみ照合します。お使いのサーバ・プラットフォームによっては大文字と小文字を区別します。
...	スラッシュを含むすべての文字に一致します。再帰的に（指定されたディレクトリおよびその配下のディレクトリ内すべてを）照合します。
%%1 - %%9	ファイル名のサブ文字列再配置に使用する位置指定子

単純なクライアント・ビュー

```
//depot/dev/... //bruno_ws/dev/...
```

では、ディポの dev ブランチ内のすべてのファイルがワークスペースの対応する場所にマッピングされます。例えば、ファイル //depot/dev/main/jam/Makefile はクライアント・ワークスペース・ファイル C:¥bruno_ws¥dev¥main¥jam¥Makefile にマッピングされます。

> **注** ファイルの不必要なマッピングを避けるため、「...」ワイルドカードの前に必ずスラッシュを入れてください。

クライアント・ワークスペース・ビュー内のマッピングは常に、ディポ内のファイルおよびディレクトリの場所を示します。クライアント・ビューにあるファイルの特定リビジョンを参照することはできません。

ディポの一部をマッピングする

ディポ・ファイルの一部分だけが必要である場合、その部分をマッピングします。クライアント・ビューの範囲を縮小することにより、コマンドが不用意に全体のディポに影響することも避けられます。クライアント・ビューを制限するには、[View:] フィールドの左側を変更し、ディポの関連部分を指定します。

例: *ディポの一部をクライアント・ワークスペースにマッピングする*

ダイは Jam プロジェクトに従事し、Web サイトを保守しているので、[View:] フィールドを次のように設定します。

```
View:
    //depot/dev/main/jam/... //dai-beos-locust/jam/...
    //depot/www/live/... //dai-beos-locust/www/live/...
```

ファイルをワークスペースの別の場所にマッピングする

ビューは複数のマッピングにより構成することができます。それを使用して、ディポ・ファイル・ツリーの一部をワークスペース・ファイル・ツリーの別の部分にマッピングできます。マッピングが衝突する場合、後のマッピングがそれ以前のマッピングに優先します。

例 : *単一のクライアント・ビューにおける複数のマッピング*

以下のビューにより、manuals フォルダにある Microsoft Word のファイルが、ワークスペース内で最上位にある wordfiles *というフォルダに格納されます。*

```
View:
    //depot/...                              //bruno_ws/...
    //depot/dev/main/docs/manuals/*.doc    //bruno_ws/wordfiles/*.doc
```

ファイルを別のファイル名にマッピングする

マッピングにより、クライアント・ワークスペース内のファイルをディポ内のファイルと異なる名前にすることもできます。

例 : *ファイルをディポ内とクライアント・ワークスペース内で異なる名前にする*

以下のビューにより、ディポ・ファイルの RELNOTES *がワークスペース・ファイルの* rnotes.txt *にマッピングされます。*

```
View:
    //depot/...                      //bruno_ws/...
    //depot/dev/main/jam/RELNOTES //bruno_ws/dev/main/jam/rnotes.txt
```

ファイル名の一部を再配置する

位置指定子の %%0 から %%9 を使用して、ファイル名およびディレクトリの一部の配置を変更することができます。

例 : *位置指定子を使用してファイル名およびディレクトリの配置を変更する*

以下のビューにより、ディポ・ファイルの //depot/allfiles/readme.txt *がワークスペース・ファイルの* /txt/readme *にマッピングされます。*

```
View:
    //depot/allfiles/%%1.%%2  //bruno_ws/filesbytype/%%2/%%1
```

ファイルおよびディレクトリを除外する

*除外マッピング*により、ファイルおよびディレクトリをクライアント・ワークスペースから明示的に除外することができます。ファイルおよびディレクトリを除外するには、マッピングの先頭にマイナス記号（-）を付けます。マイナス記号とマッピングの間にスペースを入れないように注意してください。

例 : *ビューを使用してファイルをクライアント・ワークスペースから除外する*

Jam プロジェクトに従事しているアールは、自分のワークスペースに HTML ファイルを同期したくありません。彼のクライアント・ビューは次のようになります。

```
View:
    //depot/dev/main/jam/...       //earl-dev-beech/jam/...
    -//depot/dev/main/jam/....html //earl-dev-beech/jam/....html
```

マッピングの衝突を回避する

単一のビューにおいて複数のマッピングを使用するとき、単一のファイルが不用意にディポまたはワークスペースの異なる 2 つの場所にマッピングされる場合があります。このようにして 2 つのマッピングが衝突する場合、後のマッピングがそれ以前のマッピングを無効にします。

例 : *衝突する誤ったマッピング*

ジョーは次のようにビューを構成しました。

```
View:
    //depot/proj1/...     //joe/project/...
    //depot/proj2/...     //joe/project/...
```

2 番目のマッピング //depot/proj2/... *が* //joe/project/... *にマッピングされ、最初のマッピングと衝突します。この衝突により、最初のマッピングは無視されます。* //depot/proj1 *に含まれているファイルはワークスペースにまったくマッピングされません。そのため* //depot/proj2/file.c *が存在しなくても、* //depot/proj1/file.c *はマッピングされません。*

異なるディポの場所を 1 つのワークスペースにマッピングする

オーバレイ・マッピングにより、2 つ以上のディポ・ディレクトリのファイルをクライアント・ワークスペースの同じ場所にマッピングできます。2 番目のディレクトリの内容をクライアント・ワークスペースにオーバーレイするには、マッピングの先頭にプラス記号 (+) を付けます。

例 : *複数のディレクトリを同じワークスペースにオーバーレイする*

ジョーは、ワークスペースを同期するときにプロジェクトのファイルを結合したいので、次のようにビューを構成しました。

```
View:
    //depot/proj1/...     //joe/project/...
    +//depot/proj2/...    //joe/project/...
```

オーバレイ・マッピングである +//depot/proj2/... *は* //joe/project *にマッピングし、最初のマッピングをオーバレイします。オーバレイ・マッピングは衝突しません。* //depot/proj2 *内のファイル（削除されたファイルも含む）が* //depot/proj1 *よりも優先されます。* //depot/proj2/file.c *が存在しない場合（「存在する」ファイルに相対するが、「削除されたファイル」ではない）、代わりに* //depot/proj1/file.c *がワークスペースにマッピングされます。*

オーバーレイ・マッピングは、ビルド環境においてスパース・パッチを適用する際に役立ちます。

ファイル名およびディレクトリ名におけるスペースの取り扱い

スペースを含むファイルまたはディレクトリは、引用符を使用して囲みます。

例 : *ファイル名とディレクトリ名のスペースに対応する*

ジョーは、ディポ内のファイルをローカルのワークスペースにマッピングしたいのですが、いくつかのパスにスペースが含まれています。

```
View:
    "//depot/Release 2.0/..."   //joe/current/...
    "//depot/Release 1.1/..."   "//joe/Patch Release/..."
    //depot/webstats/2006/..    "//joe/2006 Web Stats/..."
```

マッピングのサーバ側、クライアント側、または両側で、パスの構成要素の前後に引用符を挿入することにより、ジョーはスペースを含むファイル名やディレクトリ構成要素を指定することができます。

詳しくは、34 ページの「ファイル名、パス名、識別子でのスペースの使用」を参照してください。

Windows のワークスペースを複数のドライブにマッピングする

複数の Windows ドライブにわたって PERFORCE クライアント・ワークスペースを指定するには、[Root:] を null に設定し、クライアント・ビューでドライブ文字を（小文字で）指定します。以下に例を示します。

```
Client:     bruno_ws
Update:     2004/11/29 09:46:53
Access:     2005/03/02 10:28:40
Owner:      bruno
Root:       null
Options: noallwrite noclobber nocompress unlocked nomodtime normdir
SubmitOptions: submitunchanged
LineEnd: local
View:
    //depot/dev/...      "//bruno_ws/c:/Current Release/..."
    //depot/release/...  "//bruno_ws/d:/Prior Releases/..."
    //depot/www/...      //bruno_ws/d:/website/...
```

同一のワークスペースを異なるマシンで使用する

デフォルトでは、[Host:] フィールドで指定されたマシン上のワークスペースのみが使用できます。同一のクライアント・ワークスペースを異なるプラットフォームの複数のマシン上で使用したい場合、クライアント仕様の [Host:] 項目を削除し、[AltRoots:] フィールドを設定してください。クライアント・ワークスペース・ルートは 2 つまで指定できます。指定する場所は、例えば NFS または Samba マウントを介して、使用するすべてのマシンから参照可能である必要があります。

PERFORCE は現在の作業ディレクトリを最初にメインの [Root:] と比較し、次に（指定されていれば）2 つの [AltRoots:] と比較します。現在の作業ディレクトリと一致するルート・ディレクトリが使用されます。ルートが一致しない場合、メイン・ルートが使用されます。

> **注** クライアント・ルートに Windows のディレクトリを使用している場合、その Windows のディレクトリをメインのクライアント [Root:] に指定し、他のワークスペース・ルート・ディレクトリを [AltRoots:] フィールドに指定してください。

以下の例において、ユーザ bruno の現在の作業ディレクトリが /usr/bruno の下である場合、PERFORCE は c:¥bruno_ws ではなく UNIX のパスをクライアント・ワークスペース・ルートとして使用します。この手法により、bruno は UNIX と Windows の両開発環境において同一のクライアント・ワークスペース仕様を使用することができます。

```
Client: bruno_ws
Owner:  bruno
Description:
        Created by bruno.
Root:   c:\bruno_ws
AltRoots:
        /usr/bruno/
```

どちらのワークスペース・ルートが適用されているか確認するには、p4 info コマンドを発行して[Client root:]フィールドをチェックします。

同一のワークスペースにあるテキスト・ファイルを異なるプラットフォームから編集する場合、使用するエディタおよび設定で行末が保持されることを確認してください。異なるプラットフォーム間での行末コード設定について詳しくは、*『PERFORCE システム管理者ガイド』*をご覧ください。

ワークスペースの場所を変更する

ワークスペース内のファイルの場所を変更するには、p4 client コマンドを発行して[Root:]フィールドおよび[View:]フィールドの一方または両方を変更してください。これらの設定を変更する前に、(作業状態のファイルをサブミットまたは復元することにより)チェックアウトされているファイルがないようにしてください。

両方のフィールドを変更する場合、以下の手順により、ワークスペースのファイルが正しく配置されるようにしてください。

1. ファイルをワークスペース内の古い場所から削除するため、p4 sync ...#none コマンドを発行します。
2. [Root:]フィールドを変更します。
3. ファイルをワークスペース内の新しい場所にコピーするため、p4 sync コマンドを発行します。(クライアント・ビューを変更する前に p4 sync ...#none を実行するのを忘れても、後から手動でファイルを元のクライアント・ワークスペースから削除することができます。)
4. [View:]フィールドを変更します。
5. 再度 p4 sync を実行します。クライアント・ワークスペース内のファイルが新しい場所に同期されます。

ワークスペース・オプションの構成

ワークスペースの [Options:] フィールドの詳細を以下の表に示します。

オプション	意味	デフォルト
[no]allwrite	作業状態になっていないファイルは常に書き込み可能とするかを指定します。デフォルトでは、PERFORCE サーバは作業状態になっていないファイルを読み取り専用にします。ファイルの不用意な上書きや同期の失敗を防ぐため、noallwrite を指定してください。	noallwrite
[no]clobber	p4 sync の実行時、作業状態になっていない書き込み可能ファイルを上書きするかを指定します。(デフォルトでは、作業状態になっていないファイルが書き込み可能である場合、PERFORCE はそれを上書きしません。)	noclobber
[no]compress	クライアントとサーバの間で送信されるデータを圧縮するかを指定します。	nocompress
[un]locked	他のユーザによりクライアント・ワークスペース仕様の使用、編集または削除が可能であるかを指定します。PERFORCE 管理者は -f (force) オプションを使ってロックをオーバーライドすることができます。 クライアント・ワークスペース仕様をロックする場合、必ずワークスペース所有者用のパスワードを p4 passwd コマンドにより設定してください。	unlocked

オプション	意味	デフォルト
[no]modtime	+m (modtime) ファイルタイプ修飾子を***持たない***ファイルの場合 : ・ modtime が設定されている場合、ファイルがディポにサブミットされたときにファイルに表示された日付が、新たに同期された*ファイルの*更新日 (ローカル・ファイルシステム上の) になります。 ・ nomodtime が設定されている場合、PERFORCE クライアントのバージョンに関係なく、同期の日時が更新日になります。 +m (modtime) ファイルタイプ修飾子を*持つ*ファイルの場合、クライアントの modtime または nomodtime の設定に関係なく、ファイルがディポにサブミットされたときにファイルに表示された日付が、新たに同期されたファイルの更新日 (ローカル・ファイルシステム上の) になります。	nomodtime (同期の日時) +m ファイルタイプ修飾子を持つファイルについては無視されます。
[no]rmdir	ディレクトリ内のすべてのファイルが削除された場合、p4 sync によりワークスペース内の空のディレクトリを削除するかを指定します。	normdir

サブミット・オプションの設定

チェンジリストをディポにサブミットするとき、チェンジリスト内のファイルをどのように扱うかを制御するには、[SubmitOptions:] フィールドを設定します。有効な設定を以下に示します。

オプション	意味
submitunchanged	すべての作業状態のファイル (変更の有無にかかわらず) がディポにサブミットされます。これは PERFORCE のデフォルトの動作です。
submitunchanged+reopen	すべての作業状態のファイル (変更の有無にかかわらず) がディポにサブミットされ、すべてのファイルが自動的にデフォルト・チェンジリストで再度作業状態にされます。
revertunchanged	内容またはファイルタイプが変更されたファイルのみがディポにサブミットされます。未変更のファイルは元に戻します。
revertunchanged+reopen	内容またはファイルタイプが変更されたファイルのみがディポにサブミットされ、デフォルト・チェンジリストで再度作業状態にされます。未変更のファイルは元に戻しますが、デフォルト・チェンジリストでは作業状態にされません。
leaveunchanged	内容またはファイルタイプが変更されたファイルのみがディポにサブミットされます。未変更のファイルはデフォルト・チェンジリストに移動されます。
leaveunchanged+reopen	内容またはファイルタイプが変更されたファイルのみがディポにサブミットされます。未変更のファイルはデフォルト・チェンジリストに移動され、変更済みのファイルはデフォルト・チェンジリストで再度作業状態にされます。このオプションは submitunchanged+reopen と似ていますが、未変更のファイルはディポにサブミットされません。

行末コードの設定

テキスト・ファイルを同期するとき、行末をどのように変換するかを指定するには、[LineEnd:] フィールドを設定します。有効な設定を以下に示します。

オプション	意味
local	クライアントに固有のモードを使用（デフォルト）
unix	UNIX 形式（および Mac OS X）の行末識別 : LF
mac	OS X より前のバージョンの Macintosh の行末識別 : CR のみ
win	Windows 形式 : CR、LF
share	shareオプションは混在した行末をUNIXの行末形式に標準化します。shareオプションはクライアント・ワークスペースに同期されたファイルには影響しません。しかし、ファイルが PERFORCE サーバにサブミットして戻される際、share オプションにより Windows 形式の行末である CR/LF および Mac 形式の行末である CR はすべて UNIX 形式の LF に変換され、LF はそのまま残されます。 クライアント・ワークスペースの同期の際、行末は LF に設定されます。ファイルを Windows マシンで編集する場合、エディタにより各 LF の前に CR が挿入されても、追加された CR はアーカイブ・ファイル内には現れません。 share オプションをよく使用するのは、UNIX ホーム・ディレクトリをネットワーク・ドライブとしてマウントしている Windows ワークステーションのユーザでしょう。

PERFORCE による行末設定の使用方法に関して詳しくは、PERFORCE Web サイトのテクニカル・ノート 63 を参照してください。

```
http://www.perforce.com/perforce/technotes/note063.html
```

クライアント・ワークスペース仕様を削除する

クライアント・ワークスペース仕様を削除するには、`p4 client -d` *`clientname`* コマンドを使用します。クライアント仕様を削除すると、PERFORCE サーバのワークスペースについての記録が削除されますが、ワークスペースまたはディポから実際にファイルが削除されることはありません。

ワークスペース仕様を削除する手順を以下に示します。

1. ワークスペースから作業状態にされた作業中チェンジリストを元に戻すか、サブミットします。
2. クライアント・ワークスペースから既存のファイルを削除します（`p4 sync ...#none`）。（任意）
3. ワークスペース仕様を削除します。

ワークスペースからファイルを削除する前にワークスペース仕様を削除する場合、お使いのオペレーティング・システムのファイル削除コマンドを使用してワークスペースのファイルを削除できます。

セキュリティ

セキュリティを確保するため、PERFORCE 管理者は PERFORCE サーバにパスワードを設定したり、ログイン・チケットの有効期間の長さを制限することができます。設定について、次のセクションで詳しく説明します。

パスワード

PERFORCE サーバが適用しているセキュリティ・レベルによって、PERFORCE コマンドを実行する前に PERFORCE にログインする必要がある場合があります。パスワードを使用しないと、どのユーザも、P4USER の値を別のユーザ名に設定したり、-u オプションを付けて p4 コマンドを発行することにより、他の PERFORCE ユーザとして操作することが可能です。セキュリティを向上させるには、パスワードを使用してください。

パスワードの設定

PERFORCE ユーザにパスワードを作成するには、p4 passwd コマンドを発行してください。

システム管理者は、PERFORCE サーバが「強力な」パスワードを要求するように設定することができます。パスワードが 8 文字以上で、以下の条件の 2 つ以上に該当していれば、そのパスワードは強力であるといえます。

- パスワードに大文字が含まれている
- パスワードに小文字が含まれている
- パスワードに非アルファベット文字が含まれている

例えば、a1b2c3d4、A1B2C3D4、aBcDeFgH といったパスワードは強力であるとみなされます。

パスワードのリセットや削除を行うには（パスワードを知らない場合）、PERFORCE のスーパーユーザ権限が必要です。パスワードをリセットする必要が生じた場合は、PERFORCE 管理者に連絡してください。詳しくは、*『PERFORCE システム管理者ガイド』* をご覧ください。

パスワードの使用

PERFORCE ユーザにパスワードが設定されている場合、p4 コマンドを発行する際にはそのパスワードを使用しなければなりません。パスワードを使用するには、次のいずれかの方法を用います。

- 他のコマンドを発行する前に p4 login コマンドを発行し、サーバにログインする
- 環境設定または構成ファイルにおいて、P4PASSWD に自分のパスワードを設定する
- p4 コマンド発行時に -P *password* オプションを指定する（例 : p4 -P *mypassword submit*）
- Windows の場合 : p4 set -s コマンドを使用して、レジストリにパスワードを保存する。セキュリティ・レベルの高いサイトにはお奨めしません。PERFORCE 管理者はこの機能を無効にすることができます。

接続時間の制限

PERFORCE 管理者は、ユーザに対して時間制限を設定するように PERFORCE サーバを構成することができます。PERFORCE ではチケットベースの認証を使用して時間制限を設定します。チケットベースの認証は環境変数やコマンドライン・オプションに依存しないため、パスワードベースの認証より安全です。

チケットはユーザのホーム・ディレクトリ内のファイルに保存されます。チケットはログイン後、制限時間 (デフォルトでは 12 時間) が経過するまで有効です。

ログインおよびログアウト

サーバに時間制限が適用されている場合、`p4 login` コマンドを発行してチケットを取得しなければなりません。パスワード・プロンプトにパスワードを入力してください。ログインに成功すると、チケットがホーム・ディレクトリ内のチケット・ファイルに作成されます。チケットの有効期限が切れるか、`p4 logout` コマンドの発行によりログアウトするまで、再ログインは要求されません。

ログイン期限が切れるまでの時間を確認するには、次のコマンドを発行します。

```
p4 login -s
```

チケットが有効である場合、残り時間が表示されます。

PERFORCE からログアウトするには、以下のコマンドを発行します。

```
p4 logout
```

複数のマシンを使用する場合

デフォルトでは、チケットはログインしたマシンの IP アドレスに対してのみ有効です。(多くの UNIX 環境にみられるように) 1 つのホーム・ディレクトリを共有する複数のマシンから PERFORCE を使用する場合、次のコマンドによりログインします。

```
p4 login -a
```

`p4 login -a` を使用すると、チケットはすべての IP アドレスから有効なホーム・ディレクトリ内に作成されるため、PERFORCE に複数のマシンからログインすることができます。

すべてのマシンから同時にログアウトするには、次のコマンドを発行します。

```
p4 logout -a
```

p4 login および p4 logout コマンドに関して詳しくは、*『PERFORCE コマンド・リファレンス』* をご覧ください。

Unicode サーバで作業する

PERFORCE サーバを Unicode モードで起動して、Unicode 文字を含むファイル名やディレクトリ名、および Unicode 文字を含む PERFORCE 識別子 (ユーザ名など) や仕様 (チェンジリストのコメントまたはジョブ) に対するサポートを有効にすることができます。

Unicode モードの PERFORCE サーバは、Unicode のファイルおよびメタデータをクライアント・マシン用に構成された文字セットに変換し、unicode ファイルやメタデータに有効な UTF-8 の文字が含まれていることを検証します。

> **注** Unicode 文字を含むテキスト形式ファイルの管理のみが必要であり、上記の機能が必要でない場合は、サーバを Unicode モードで起動する必要はありません。お使いのサーバが Unicode モードを使用しているかどうかは、システム管理者にお尋ねください。
>
> そのようなインストールでは、Unicode 文字を含むテキスト形式ファイルに Perforce utf16 というファイルタイプを割り当ててください。詳しくは、106 ページの「Unicode のファイルにファイルタイプを割り当てる」を参照してください。

Unicode モードのサーバ間で適切に相互運用を行うため、またファイルの同期やサブミットの際にそのようなファイルが PERFORCE サーバによって正しく変換されるように、P4CHARSET をテキストエディタや IDE などそれらにアクセスするアプリケーションによってクライアント・マシンで使用される形式に対応した文字セットに設定しなければなりません。これらの形式は通常、**[名前を付けて保存 ...]** メニュー・オプションを使用してファイルを保存する際に一覧表示されます。

utf16 または utf32 で始まる P4CHARSET の値については、さらに P4COMMANDCHARSET をサーバ出力を表示させたい utf16 または utf32 以外の文字セットに設定する必要があります。「サーバ出力」には、情報やエラーのメッセージ、差分出力、およびレポート作成コマンドにより返される情報が含まれます。

詳しくは *『システム管理者ガイド』* を参照してください。

Windows 環境での P4CHARSET の設定

あるワークステーションのすべてのユーザに P4CHARSET を設定するには、PERFORCE 管理者権限が必要です。次のコマンドを実行してください。

```
p4 set -s P4CHARSET=character_set
```

現在ログインしているユーザに P4CHARSET を設定するには、

```
p4 set P4CHARSET=character_set
```

お使いのクライアントマシンに、True Type または Open Type のフォントがインストールされている必要があります。

UNIX 環境での P4CHARSET の設定

コマンドシェルまたは .kshrc、.cshrc、.profile などの起動スクリプトに P4CHARSET を設定することができます。P4CHARSET の適切な値を決定するには、LANG または LOCALE の環境変数の設定を調べます。一般的な設定を以下に示します。

LANG の設定	P4CHARSET の設定
en_US.ISO_8859-1	iso8859-1
ja_JP.EUC	eucjp
ja_JP.PCK	shiftjis

通常は、日本語のインストールでは P4CHARSET を eucjp に設定し、ヨーロッパ言語のインストールでは P4CHARSET を iso8859-1 に設定します。

第 3 章

P4 コマンドの発行

本章では p4 コマンドについての基本的な情報を、コマンドライン構文、引数、オプションも含めて説明します。コマンドの構文に関する詳細説明は、*『PERFORCE コマンド・リファレンス』*をご覧ください。

いくつかのコマンドには管理者権限またはスーパーユーザ権限が必要です。詳しくは、*『PERFORCE システム管理者ガイド』*をご覧ください。

コマンドラインの構文

コマンドの基本構文は以下のとおりです。

p4 [*global options*] *command* [*command-specific flags*] [*command arguments*]

以下のオプションはすべての p4 コマンドで使用できます。

グローバル・オプション	説明および使用例
-c *clientname*	コマンドに関連するクライアント・ワークスペースを指定します。P4CLIENT をオーバーライドします。
	p4 -c bruno_ws edit //depot/dev/main/jam/Jambase
-d *directory*	環境変数 PWD の設定をオーバーライドし、現在のディレクトリを指定します。
	p4 -d ~c:¥bruno_ws¥dev¥main¥jam¥Jambase Jamfile
-G	すべての出力を整列された Python ディレクトリ・オブジェクトとしてフォーマットします（Python によるスクリプト作成用）。
	p4 -G info
-H *host*	P4HOST の設定をオーバーライドし、クライアント・ワークステーションのホスト名を指定します。
	p4 -H deneb print //depot/dev/main/jam/Jambase
-p *server*	P4PORT の設定をオーバーライドし、PERFORCE サーバのホストおよびポート番号を指定します。
	p4 -p deneb:1818 clients
-P *password*	P4PASSWD の設定をオーバーライドし、パスワードを指定します。通常、-u *username* オプションと組み合わせて使用します。
	p4 -u earl -P secretpassword job
-s	出力の各行の先頭に、タグを付加します（スクリプト作成用）。
	p4 -s info
-u *username*	P4USER の設定をオーバーライドし、PERFORCE ユーザ名を指定します。
	p4 -u bill user

グローバル・オプション	説明および使用例
-x *filename*	指定したファイルから 1 行につき 1 つの引数を読み取ります。引数を標準入力から読み取るには、"-x -" と指定します。 p4 -x myargs.txt
-V	p4 実行プログラムのバージョンを表示します。

特定のコマンドのオプションを表示するには、p4 help コマンドを発行します。例を示します。

```
p4 help add
    add -- Open a new file to add it to the depot
    p4 add [ -c changelist# ] [ -t filetype ] file ...
        Open a new file for adding to the depot. If the file exists
        on the client it is read to determine if it is text or binary.
        If it does not exist it is assumed to be text. The file must
        either not exist in the depot, or it must be deleted at the
        current head revision. Files may be deleted and re-added.
[...]
```

グローバル・オプション、コマンド、コマンド固有のオプションについての完全なリストは、*『PERFORCE コマンド・リファレンス』* をご覧ください。

コマンドラインでファイル名を指定する

PERFORCE の日常操作の大部分は、ファイルの管理です。p4 コマンドでは以下のようにファイル名を指定できます。

- **ローカル・シンタックス** : ローカル・シェルまたは OS で指定されるファイル名。

  ファイル名は、絶対パス（例 : c:¥bruno_ws¥dev¥main¥jam¥fileos2.c）または現在のディレクトに対する相対パス（例 : .¥jam¥fileos2.c）により指定することができます。

  相対パスのコンポーネント (. または ..) を固定パスのコンポーネントの後に指定することはできません。例えば、mysub/mydir/./here/file.c は mysub/mydir という固定コンポーネントの後にドット (.) が指定されているため、無効です。

- **ディポ・シンタックス** : 使用されるフォーマットは *//depotname/file_path* で、ファイルのパス名をディポのルート・ディレクトリに相対して指定します。パスのコンポーネントは、スラッシュで区切られます。例 : //depot/dev/main/jam/JambaseÅB

- **クライアント・シンタックス**: f使用されるフォーマットは *//workspacename/file_path* で、ファイルのパス名をクライアント・ルート・ディレクトリに相対して指定します。パスのコンポーネントは、スラッシュで区切られます。
  例 : //ona-agave/dev/main/jam/Jambase。

例 : *同じファイルを異なるシンタックスにより参照する*

ローカル・シンタックス : p4 delete c:¥bruno_ws¥dev¥main¥jam¥Jambase

ディポ・シンタックス : p4 delete //depot/dev/main/jam/Jambase

クライアント・シンタックス : p4 delete //bruno_ws/dev/main/jam/Jambase

PERFORCE のワイルドカード

ファイルの集合を操作するコマンドのために、PERFORCE では 2 種類のワイルドカードがサポートされています。

ワイルドカード	説明
*	スラッシュを除くすべての文字に一致します。単一のディレクトリ内のみ照合します。お使いのサーバ・プラットフォームによっては大文字と小文字を区別します。
...	スラッシュを含むすべての文字に一致します。再帰的に（指定されたディレクトリおよびその配下のディレクトリ内すべてを）照合します。

PERFORCE ワイルドカードは、以下の例のようにローカル・シンタックス、PERFORCE シンタックスの両方で使用できます。

表記	意味
J*	J で始まるカレント・ディレクトリ内のファイル
*/help	現サブディレクトリ内の help という名のファイルすべて
./...	カレント・ディレクトリとそのサブディレクトリに属する全ファイル
./....c	カレント・ディレクトリとそのサブディレクトリに属し、.c で終わる全ファイル
/usr/bruno/...	/usr/bruno に属する全ファイル
//bruno_ws/...	bruno_ws という名前のワークスペース内またはディポ内の全ファイル
//depot/...	ディポ内の全ファイル
//...	全ディポ内の全ファイル

ワイルドカード * は、コマンドがサーバに送られる前にローカル OS により拡張されます。ローカル OS によってワイルドカード * が拡張されないようにするには、* を引用符で囲むか、前に逆スラッシュを付けます。

> **注** *ワイルドカード "..." は p4 add コマンドでは使用できません。ワイルドカード "..." は PERFORCE サーバによって拡張されますが、サーバでは追加されるファイルを判断できないため、ワイルドカードを拡張できません。ワイルドカード * は、PERFORCE サーバではなく OS シェルにより拡張されるため、*p4 add *で使用できます。

ファイル名と識別子における制限

ファイル名、パス名、識別子でのスペースの使用

スペースを含むファイルやディレクトリは引用符で囲み、例えば次のように指定します。

```
"//depot/dev/main/docs/manuals/recommended configuration.doc"
```

その他の PERFORCE オブジェクト、例えばブランチ名、クライアント名、ラベル名などの中にスペースを指定すると、そのスペースは PERFORCE サーバによって自動的にアンダースコアに変換されます。

長さの制限

ブランチやクライアント・ワークスペースなどの PERFORCE オブジェクトに付ける名前は、1024 文字を超えてはなりません。

予約文字

デフォルトでは、以下に示す予約文字を PERFORCE 識別子または PERFORCE により管理されるファイルの名前に使用することはできません。

予約文字	理由
`@`	日付、ラベル名、チェンジリスト番号に使用するファイル・リビジョン指定子
`#`	ファイル・リビジョン番号
`*`	ワイルドカード
`...`	ワイルドカード（再帰的）
`%%1 - %%9`	ワイルドカード（位置的）
`/`	パス名コンポーネント分離記号

上記の文字には、衝突する別の用法があります。その衝突には以下のものが含まれます。

- UNIX ではパス・コンポーネントを / で区切りますが、DOS コマンドでは大抵、/ をコマンドラインスイッチと解釈します。
- UNIX シェルではほとんどの場合、# をコメントの始まりと解釈します。
- UDOS シェル、UNIX シェルの両方とも、複数のファイルと照合するよう * を自動的に拡張し、DOS コマンドラインでは % により変数を表します。

これらの文字をファイル名やパス名に指定するには、以下に示す ASCII 文字表現の 16 進値を使用します。

文字	ASCII 表現
@	%40
#	%23
*	%2A
%	%25

ファイルを追加する際に字義通りにファイル名を指定し、後に ASCII 拡張を使用してそのファイル名を参照します。例えば、recommended@configuration.doc というファイルを追加するには、以下のコマンドを発行します。

```
p4 add -f
//depot/dev/main/docs/manuals/recommended@configuration.doc
```

チェンジリストをサブミットすると、文字は自動的に拡張され、変更サブミット・フォームに次のように表示されます。

```
//depot/dev/main/docs/manuals/recommended%40configuration.doc
```

ファイル追加を含むチェンジリストをサブミットした後、ASCII 拡張を使用して、ファイルをワークスペースに同期させるか、ワークスペース内で編集する必要があります。例えば次のようにします。

```
p4
sync //depot/dev/main/docs/manuals/recommended%40configuration.doc
```

拡張（非 ASCII）文字を含むファイル名

ファイル名および PERFORCE 識別子には非 ASCII 文字を使用できますが、これらの文字をコマンドラインから入力するには、プラットホーム特有の解決法が必要な場合があります。PERFORCE を Unicode モードで使用する場合、すべてのユーザが P4CHARSET を適切に設定していなければなりません。P4CHARSET の設定に関して詳しくは、*『PERFORCE コマンド・リファレンス』* および *『PERFORCE 国際語モードに関する注意事項』* をご覧ください。

国際語化環境では、共通コード・ページまたはロケール設定を使用して、組織内のすべてのマシン間ですべてのファイル名が正しく表示されるようにしてください。コード・ページまたはロケールを設定するには、次のようにします。

- Windows の場合は、[**コントロールパネル**] の [**地域設定**] アプレットを使用してください
- UNIX の場合は、環境変数 LOCALE を設定してください。

ファイル・リビジョンを指定する

ファイルをディポにサブミットするたびに、リビジョン番号は 1 つずつ増加します。最新リビジョンより前のリビジョンを指定するには、リビジョン指定子 # を使用してリビジョン番号を指定するか、@ を使用して作業中のファイルのバージョンに対応する日付、チェンジリスト、クライアント・ワークスペースまたはラベルを指定します。リビジョン仕様を使用して、コマンドの有効範囲を特定のファイル・リビジョンに制限することができます。

> **警告** *OS シェルの中には、PERFORCE リビジョン・キャラクタ # が語の最初にある場合、# をコメント・キャラクタとして処理するものがあります。ご使用のシェルがこれに該当する場合は、p4 コマンドで使用する際に # をエスケープしてください。*

以下の表に、ファイル・リビジョンの種々の指定方法を示します。

対象リビジョン	構文と例
リビジョン番号	*file*#*n* 例 : `p4 sync //depot/dev/main/jam/Jambase#3` ファイル Jambase のリビジョン 3 を参照

特定のチェンジリストの時点でサブミットしたリビジョン	*file*@*changelist_number*
	例 :
	`p4 sync //depot/dev/main/jam/Jambase@126`
	チェンジリスト 126 がサブミットされたときの Jambase のバージョンを参照。Jambase への変更がチェンジリスト 126 でサブミットされなかった場合も有効。
	`p4 sync //depot/...@126`
	チェンジリスト 126 におけるディポ全体の状態を参照（番号付チェンジリストの解説は 46 ページの「チェンジリストの管理」に記載）。

特定のラベルにおけるリビジョン	*file*@*labelname* 例 : `p4 sync //depot/dev/main/jam/Jambase@beta` beta というラベルが付けられた Jambase のリビジョン。ラベルの詳細情報は、71 ページの「ラベルの使用」に記載。
特定のクライアント・ワークスペースに最後に同期されたリビジョン	*file*@*clientname* 例 : `p4 sync //depot/dev/main/jam/Jambase@bruno_ws` クライアント・ワークスペース bruno_ws に最後に同期された Jambase のリビジョン。
ファイルを削除	*file*#none 例 : `p4 sync //depot/dev/main/jam/Jambase#none` Jambase をクライアント・ワークスペースから削除。
ファイルの最新リビジョン	*file*#head 例 : `p4 sync //depot/dev/main/jam/Jambase#head` `p4 sync //depot/dev/main/jam/Jambase` と同じ。（リビジョン指定子を省略すると、head リビジョンが同期されます。）

対象リビジョン	構文と例
ワークスペースに最後に同期されたリビジョン	*file*#have 例 : p4 files //depot/dev/main/jam/Jambase#have
特定の日におけるディポ内のファイルの最新リビジョン	*file*@*date* 例 : p4 sync //depot/dev/main/jam/Jambase@2005/05/18 2005 年 5 月 18 日午前 0 時現在の Jambase の最新リビジョン。
特定の日の特定時刻におけるディポ内のファイルの最新リビジョン	*file*@"*date*[:*time*]" 例 : p4 sync //depot/dev/main/jam/Jambase@”2005/05/18” 日付を *YYYY/MM/DD* の形式で指定し、時刻を 24 時間表記で *HH:MM:SS* の形式で指定します。時刻のデフォルトは 00:00:00 です。 日付と時刻を単一のスペースまたはコロンで区切ります（スペースで区切る場合、日付と時刻の指定全体を二重引用符で囲む必要があります)。

例 : *リビジョン指定子を使ってファイルを取得する*

ブルーノはチェンジリスト番号 30 に存在していたすべてのリビジョンを取得したいので、次のように入力します。

```
p4 sync //depot/dev/main/jam/Jambase@30
```

別のユーザが自分のワークスペースに、ブルーノが同期させたものと同じファイル・リビジョンを同期させることができます。それには、ブルーノのワークスペースを次のように指定します。

```
p4 sync @bruno_ws
```

例 : *クライアント・ワークスペースからすべてのファイルを削除する*

```
p4 sync ...#none
```

ファイルはワークスペースから削除されますが、ディポからは削除されません。

日付と時刻の表記

日付と時刻の表記は、PERFORCE サーバのタイムゾーンから取得されます。ご使用の PERFORCE サーバで有効な日付、時刻、グリニッジ標準時（GMT）からのオフセットおよびタイムゾーンを表示するには、p4 info コマンドを発行してください。PERFORCE サーバは、時刻をグリニッジ標準時の 1970 年 1 月 1 日 00 時 00 分 00 秒以降の秒数として表します。したがって、タイムゾーンをまたがってサーバを移動させると、サーバに保存された時刻は新しいタイムゾーンで正しく報告されます。

リビジョン範囲

PERFORCE コマンドには、ファイル・リビジョンの範囲を限定して実行できるものがあります。リビジョン範囲を指定するには、開始リビジョンと終了リビジョンをカンマで区切り、#3、4 のように指定します。

リビジョン範囲の指定が可能なコマンドを以下に示します。

- `p4 changes`
- `p4 files`
- `p4 integrate`
- `p4 jobs`
- `p4 print`
- `p4 sync`

上記のコマンドにおいて、

- 単一のリビジョンを指定する場合、コマンドの実行範囲は #1 から指定したリビジョンまでになります（ただし、`p4 sync`、`p4 print`、`p4 files` を除きます。それらのコマンドでは範囲内の最新のリビジョンが処理されます)。
- リビジョン範囲を完全に省略すると、コマンドはすべてのファイル・リビジョンに作用します。

例: *リビジョン範囲を指定してチェンジを一覧表示する*

リリース・マネージャは、2000 年 7 月の jam プロジェクトに対するすべてのチェンジの一覧表を見る必要があります。マネージャは次のように入力します。

```
p4 changes //depot/dev/main/jam/...@2000/7/1,2000/8/1
```

生成されるチェンジのリストは次のとおりです。

```
Change 673 on 2000/07/31 by bruno@bruno_ws 'Final build for QA'
Change 633 on 2000/07/1 by bruno@bruno_ws 'First build w/bug
fix'
Change 632 on 2000/07/1 by bruno@bruno_ws 'Started work'
```

レポート・コマンド

有用なレポート・コマンドを以下の表に示します。

表示内容	使用するコマンド
すべての p4 コマンドを、短い説明付きでリストします。	`p4 help commands`
与えられたコマンド（*command*）についての詳細なヘルプを示します。	`p4 help command`
すべての PERFORCE コマンドに共通のコマンドライン・オプションを記述します。	`p4 help usage`
PERFORCE ビュー構文について説明します。	`p4 help views`
p4 help コマンドに与えられるすべての引数について記述します。	`p4 help`
クライアント・マシンに対して構成された PERFORCE の設定	`p4 info`
クライアント・ワークスペース内のファイル・リビジョン	`p4 have`
p4 sync の実行結果をプレビューします（転送されるファイルを確認するため）。	`p4 sync -n`
p4 delete の実行結果をプレビューします（削除目的でマーキングされるファイルを確認するため）。	`p4 delete -n files`

PERFORCE フォームの使用

PERFORCE コマンドのうち、p4 client や p4 submit などは、テキストエディタを使用してフォームを表示し、ユーザがそのフォームからコマンドを完了させるのに必要な情報（例えばサブミットするチェンジの説明など）を入力します。ユーザがフォームを変更し、保存してエディタを終了すると、PERFORCE はフォームを解析し、フォーム内の情報を使用してコマンドを完了させます。(PERFORCE フォームの表示および編集に使用されるテキストエディタを指定するには、P4EDITOR を設定します。)

PERFORCE フォームに情報を入力する際は、次の規則に従ってください。

- フィールド名（[View:] など）は左づめ（インデントなし）で、末尾にコロンを付けます。
- フィールド値（入力値）はフィールド名と同じ行に記述するか、フィールド名の下の行にタブでインデントを付けて記述します。

いくつかのフィールド名、例えば p4 client フォームの [CLIENT:] (クライアント名 :) フィールドなどは、単一の値になります。他のフィールド、例えば [Description:] (コメント :) などは、テキスト・ブロックの形式になります。さらに他のフィールド、例えば [View:] などは、1 行に複数の値を取ります。

クライアント・ワークスペース・フォームの [Client:] フィールドなどの値は変更できません。他のフィールド、例えば p4 submit の [Description:] などは、値を変更*しなければなりません*。変更する必要のあるフィールドを変更しない場合、または変更不可の値を変更しようとすると、PERFORCE によりエラーが表示されます。変更可能なフィールドに関して詳しくは、*『PERFORCE コマンド・リファレンス』*を参照するか、p4 help *コマンド*を使用してください。

第 4 章

ファイルとチェンジリストの管理

この章では、ファイル管理の方法およびチーム開発環境での作業方法を説明します。チーム開発環境では、同一のファイルを複数のユーザが操作するため、各ユーザによる変更の調整が必要な場合があります。

ファイルの管理

ディポ（ファイル格納場所）内のファイルを変更するには、ファイルをチェンジリストで作業状態にし、変更についてのコメントを入力してチェンジリストをサブミットします。PERFORCE はチェンジリストに番号を割り当て、ユーザのファイルのリビジョン履歴を保存します。この方法によって、関連する変更をグループ化したり、だれがいつ、何のために変更したかを調べることができます。ファイルの操作の基本的な手順を以下に示します。

タスク	説明
同期（ファイルをディポから取得する）	ディポから取得したいファイルとディレクトリを指定して、`p4 sync` コマンドを発行します。クライアント・ビューにマッピングされているファイルのみ、同期が可能です。
ファイルをディポに追加する	1. ファイルをワークスペースで作成します。 2. ファイルをチェンジリストで追加目的の作業状態にします（`p4 add`）。 3. チェンジリストをサブミットします（`p4 submit`）。
ファイルを編集して変更をサブミットする	1. 必要に応じて、目的のファイル・リビジョンをワークスペースに同期します（`p4 sync`）。 2. ファイルをチェンジリストで編集目的の作業状態にします（`p4 edit`）。 3. ファイルを変更します。 4. チェンジリストをサブミットします（`p4 submit`）。変更を破棄するには、`p4 revert` コマンドを発行します。
ファイルをディポから削除する	1. ファイルをチェンジリストで削除目的の作業状態にします（`p4 delete`）。ファイルがワークスペースから削除されます。 2. チェンジリストをサブミットします（`p4 submit`）。ファイルはディポから削除されます。

タスク	説明
変更を破棄する	ファイルまたはファイルが作業状態になっているチェンジリストをキャンセルします。キャンセル操作は、作業状態のファイルに次のように影響します。 追加 : 影響なし－ファイルはワークスペースに残ります。 編集 : 作業状態にしたリビジョンが再度ディポから同期され、ワークスペースにあるファイルに加えた変更は上書きされます。 削除 : ファイルは再度ワークスペースから同期されます。

ディポに対するファイルの追加、削除、更新は、そのファイルが作業状態にされている作業中チェンジリストを正常にサブミットした場合にのみ行われます。1 つのチェンジリストには追加目的、編集目的、削除目的の作業状態のファイルを混在させることができます。

コマンドライン上でのファイルの指定に使用する構文について詳しくは、32 ページの「コマンドラインでファイル名を指定する」を参照してください。以降のセクションでは、ファイルの操作について詳しく説明します。

ファイルの同期（取得）

ファイルをディポからクライアント・ワークスペースに取得するには、p4 sync コマンドを発行します。自分のクライアント・ビューに存在しないファイルを同期させることはできません。クライアント・ビューの指定について詳しくは、19 ページの「クライアント・ビューの限定」をご覧ください。

例 : *ディポからクライアント・ワークスペースへのファイルのコピー*

以下のコマンドにより、クライアント・ビューにあるすべてのファイルの最新リビジョンがディポからワークスペースに取得されます。ファイルが同期されると、コマンド出力にリスト表示されます。

```
C:¥bruno_ws>p4 sync
//depot/dev/main/bin/bin.linux24x86/readme.txt#1 - added as
c:¥bruno_ws¥dev¥main¥bin¥bin.linux24x86¥readme.txt
//depot/dev/main/bin/bin.ntx86/glut32.dll#1 - added as
c:¥bruno_ws¥dev¥main¥bin¥bin.ntx86¥glut32.dll
//depot/dev/main/bin/bin.ntx86/jamgraph.exe#2 - added as
c:¥bruno_ws¥dev¥main¥bin¥bin.ntx86¥jamgraph.exe
[...]
```

p4 sync コマンドにより、クライアント・ワークスペース内のファイルが追加、更新、または削除され、ワークスペースの内容がディポと同期されます。あるファイルがディポ内の特定のサブディレクトリに存在し、そのディレクトリがクライアント・ワークスペースに存在しない場合、ディレクトリはファイルを同期するときにクライアント・ワークスペースに作成されます。ファイルがディポから削除されている場合、p4 sync コマンド実行時にクライアント・ワークスペースからも削除されます。

ディポ内のファイルの最新リビジョンより前のリビジョンを同期させるには、リビジョン指定子を使用します。例えば、複数リビジョンのある Jamfile の 1 番目のリビジョンと同期させるには、次のコマンドを発行します。

```
p4 sync//depot/dev/main/jam/Jamfile#1
```

より詳しくは、36 ページの「ファイル・リビジョンを指定する」をご覧ください。

ファイルの集合またはディレクトリ全体を同期させるには、ワイルドカードを使用します。例えば、「jam」フォルダとその配下にあるすべてのファイルを同期させるには、次のコマンドを発行します。

```
p4 sync //depot/dev/main/jam/...
```

より詳しくは、33 ページの「Perforce のワイルドカード」をご覧ください。

PERFORCE サーバでは、(サーバ・マシンにある データベースに) ユーザが同期させたリビジョンを記録しています。効率化を図るため、PERFORCE は既に同期済みのファイル・リビジョンを再度同期しません。手動で (おそらく不用意に) 削除したファイルを再度同期するには、p4 sync の発行時に -f オプションを指定してください。

ファイルの追加

ファイルをディポに追加するには、ワークスペースでファイルを作成し、p4 add コマンドを発行します。p4 add コマンドにより、ファイルはデフォルト作業中チェンジリストで追加目的の作業状態にされます。デフォルト作業中チェンジリストのサブミットが成功したとき、ファイルが追加されます。複数のファイルを追加目的の作業状態にするには、単一の p4 add コマンドでワイルドカードを使用します。ファイルを再帰的に追加するために PERFORCE の「...」ワイルドカードを使用することはできません。

プラットフォーム固有の再帰的なファイル追加 (サブディレクトリ内のファイル追加) に関して詳しくは、PERFORCE Web サイトの Tech Note 12 を参照してください。

```
http://www.perforce.com/perforce/technotes/note012.html
```

例 : *ファイルをチェンジリストに追加する*

ブルーノはディポに追加する必要のある 2、3 のテキスト・ファイルを作成しました。すべてのテキスト・ファイルを一度に追加するために、ブルーノは p4 add *コマンドの発行時に「**」*ワイルドカードを使います。*

```
C:¥bruno_ws¥dev¥main¥docs¥manuals>p4 add *.txt
//depot/dev/main/docs/manuals/installnotes.txt#1 - opened for add
//depot/dev/main/docs/manuals/requirements.txt#1 - opened for add
```

これで、ブルーノがディポに追加したいファイルが彼のデフォルト・チェンジリスト内で作業状態になりました。チェンジリストがサブミットされるとき、ファイルはディポに格納されます。

例 : *チェンジリストをディポにサブミットする*

ブルーノはファイルをディポに追加する準備ができました。彼が p4 submit *と入力すると、標準のテキスト・エディタに次のようなフォームが表示されます。*

```
Change:   new
Client:   bruno_ws
User:   bruno
Status:   new
Description:
        <enter description here>
Files:
    //depot/dev/main/docs/manuals/installnotes.txt   # add
    //depot/dev/main/docs/manuals/requirements.txt   # add
```

*ブルーノは [*Description:*] フィールドの内容を変更し、自分が行ったファイル更新の説明を記述します。入力後、フォームを保存してエディタを終了すると、新しいファイルがディポに追加されます。*

[Description:] フィールドにはコメントを入力しなければなりません。[Files:] フィールドからは行を削除しても構いません。このリストから削除されたファイルは、次のデフォルト・チェンジリストに移動され、次回デフォルト・チェンジリストをサブミットしたときにリスト表示されます。

ディポに存在しないディレクトリにファイルを追加すると、チェンジリストのサブミットが成功したときにディポのディレクトリが作成されます。

ファイルの変更

ファイルを編集目的で作業状態にするには、p4 edit コマンドを発行します。ファイルを編集目的の作業状態にすると、PERFORCE はユーザのワークスペースのファイルに対し書き込み許可を与え、そのファイルをチェンジリストに追加します。ファイルがディポには存在してもユーザのワークスペースには存在しない場合、編集目的の作業状態にする前にそのファイルを同期させる必要があります。また、ファイルを編集する場合、編集目的の作業状態にしなければなりません。

例 : *ファイルを編集目的で作業状態にする*

ブルーノは command.c *に変更を加えたいので、そのファイルを同期させ、ファイルを編集目的で作業状態にします。*

```
p4 sync //depot/dev/command.c
//depot/dev/command.c#8 - added as c:¥bruno_ws¥dev¥command.c

p4 edit //depot/dev/command.c
//depot/dev/command.c#8 - opened for edit
```

そして、ブルーノは任意のテキスト・エディタでファイルを編集します。編集後、彼は上記のように p4 submit *によりファイルをディポにサブミットします。*

変更の破棄（元に戻す）

作業状態にしたファイルをチェンジリストから削除して、それまで行った変更を破棄するには、p4 revert コマンドを発行します。ファイルを元に戻すと、PERFORCE サーバは最後にワークスペースに同期されたバージョンをリストアします。追加目的で作業状態にしたファイルを元に戻す場合、ファイルはチェンジリストからは削除されますがワークスペースからは削除されません。

例: *ファイルを元に戻す*

ブルーノは結局、自分のテキスト・ファイルを追加しないことにしました。

```
C:¥bruno_ws¥dev¥main¥docs¥manuals>p4 revert *.txt
//depot/dev/main/docs/manuals/installnotes.txt#none - was add,
abandoned
//depot/dev/main/docs/manuals/requirements.txt#none - was add,
abandoned
```

実際にはファイルを元に戻さずにキャンセル操作の結果をプレビューするには、p4 revert コマンドの発行時に -n オプションを指定します。

ファイルの削除

ファイルをディポから削除するには、p4 delete コマンドを発行して削除対象ファイルを作業状態にし、それらが作業状態にされているチェンジリストをサブミットします。ファイルをディポから削除しても以前のリビジョンは存在したままで、「削除済み」とマークされた新しい最新リビジョンが追加されます。したがってファイルの古いリビジョンを同期させることは依然として可能です。

p4 delete コマンドを発行すると、ファイルはワークスペースからは削除されますがディポからは削除されません。削除目的で作業状態にされたファイルを復元すると、それらはワークスペースに復元されます。ファイルが作業状態にされたチェンジリストのサブミットが成功すると、ファイルはディポから削除されます。

例: *ファイルをディポから削除する*

*ブルーノは次のようにして、*vendor.doc *をディポから削除します。*

```
p4 delete //depot/dev/main/docs/manuals/vendor.doc
//depot/dev/main/docs/manuals/vendor.doc#1 - opened for delete
```

ファイルはただちにクライアント・ワークスペースから削除されますが、ブルーノが p4 submit *コマンドを発行するまでディポからは削除されません。*

チェンジリストの管理

ディポ内のファイルを変更するには、それらのファイルを*チェンジリスト*で作業状態にし、ファイルを変更してからチェンジリストを*サブミット*します。チェンジリストには、ファイル、そのリビジョン番号、およびファイルに対して行われる操作のリストが含まれています。サブミットされていないチェンジリストは、*作業中チェンジリスト*と呼ばれます。

チェンジリストのサブミットは、すべて行うか全く行わないかの操作になります。つまり、チェンジリスト内のすべてのファイルがディポで更新されるか、または、エラーが発生した場合にはどのファイルも一切更新されません。この方法により、複数のファイルに影響するコード変更が同時に行われるようになっています。

PERFORCE は、チェンジリストに番号を割り当てるほか、*デフォルト・チェンジリスト*を管理します。デフォルト・チェンジリストにはサブミットの際に番号が付けられます。複数のチェンジリストを作成して作業を整理することができます。例えば、あるチェンジリストには新機能を実装するために変更されるファイルを入れ、別のチェンジリストにはバグ修正を含めることもできます。ファイルを作業状態にすると、コマンドラインで `-c` オプションを使用して既存のチェンジリスト番号を指定しない限り、そのファイルはデフォルト・チェンジリストに置かれます。例えば、チェンジリスト番号 4 でファイルを編集し、サブミットするには、`p4 edit -c 4 filename` を使用します。ファイルをデフォルト・チェンジリストで作業状態にするには、`-c` オプションを省略します。

他のユーザの操作内容によって、チェンジリストのサブミットの際に、PERFORCE サーバがチェンジリストの番号を付け替える場合があります。チェンジリストの番号が付け替えられた場合、元の番号が別のチェンジリストに割り当てられることはありません。

チェンジリストに対してファイルの追加または削除を行うコマンドを以下に示します。

- `p4 add`
- `p4 delete`
- `p4 edit`
- `p4 integrate`
- `p4 reopen`
- `p4 revert`

番号付チェンジリストをサブミットするには、`p4 submit` コマンド発行時に `-c` オプションを指定します。デフォルト・チェンジリストをサブミットするには、`-c` オプションを省略します。詳しくは、*『PERFORCE コマンド・リファレンス』*の `p4 submit` コマンドに関する説明を参照してください。

ファイルを 1 つのチェンジリストから別のチェンジリストに移動させるには、`p4 reopen -c` *`changenum filenames`* コマンドを、*`changenum`* に移動先のチェンジリストの番号を指定して発行してください。ファイルをデフォルト・チェンジリストに移動させる場合は、`p4 reopen -c default` *`filenames`* を使用します。

番号付チェンジリストの作成

番号付チェンジリストを作成するには、p4 change コマンドを発行します。このコマンドによりチェンジリスト・フォームが表示されます。コメントを入力し、必要な変更を行ってからフォームを保存し、エディタを終了します。

デフォルト・チェンジリストで作業状態になっているすべてのファイルが新しいチェンジリストに移動します。テキスト・エディタを終了すると、チェンジリストに番号が割り当てられます。ファイルをこのチェンジリストから削除すると、それらのファイルはデフォルト・チェンジリストに戻ります。

例 : *複数のチェンジリストを使って作業する*

ブルーノは 2 つのバグ修正を行っており、それぞれの修正を別のチェンジリストにサブミットする必要があります。そこで 1 番目の修正に対するファイルの最新リビジョンを同期させ、デフォルト・チェンジリストで編集目的で作業状態にします。

```
C:¥bruno_ws¥>p4 sync //depot/dev/main/jam/*.c
[list of files synced...]

C:¥bruno_ws>p4 edit //depot/dev/main/jam/*.c
[list of files opened for edit...]
```

ここでブルーノは p4 change *コマンドを発行して、チェンジリスト・フォームにコメントを入力します。彼がファイルを保存してエディタを終了させると、*PERFORCE *はファイルを含む番号付チェンジリストを作成します。*

```
C:¥bruno_ws¥dev¥main¥docs¥manuals>p4 change

    [Enter description and save form]

Change 777 created with 33 open file(s).
```

*ブルーノは 2 番目のバグ修正に対しても同様に、*p4 sync、p4 edit、p4 change *を実行します。このようにして、各修正に対して 1 つずつ、2 つの番号付チェンジリストが作成されました。*

サブミット済チェンジリストに割り当てられる番号は、チェンジリストがサブミットされた順番を表します。チェンジリストがサブミットされるとき、以下の例で示すように PERFORCE サーバが番号を付け替える場合があります。

例 : *チェンジリスト番号の自動付け替え*

ブルーノは、チェンジリスト 777 を使って作業していたバグの修正を完了しました。そのチェンジリストを作成した後に、ブルーノは別のチェンジリストをサブミットし、他の 2 人のユーザもチェンジリストをサブミットしました。

ブルーノが p4 submit -c 777 *を実行してチェンジリスト 777 をサブミットすると、次のようなメッセージが表示されます。*

```
Change 777 renamed change 783 and submitted.
```

チェンジリストのサブミット

作業中チェンジリストをサブミットするには、p4 submit コマンドを発行します。p4 submit コマンドを発行すると、フォームが表示され、そこにチェンジリストにあるファイルがリストされます。このリストからファイルを取り除くこともできます。取り除いたファイルは、サブミットまたは復元を行わない限り、デフォルトの作業中チェンジリスト内に作業状態のまま残ります。

デフォルト・チェンジリストで作業状態にされている特定のファイルをサブミットするには、p4 submit *filename* コマンドを発行します。ファイルの集合を指定するには、ワイルドカードを使用します。例えば、デフォルト・チェンジリストで作業状態にされているすべてのテキストファイルをサブミットするには、p4 submit "*".txt と入力します。(エスケープ・コードとして引用符を * ワイルドカードの前後に付けて、ローカル・コマンド・シェルにより解釈されないようにしてください。)

チェンジリスト・フォームを保存してテキスト・エディタを終了すると、チェンジリストが PERFORCE サーバにサブミットされ、サーバがディポのファイルを更新します。チェンジリストのサブミットが成功すると、それを変更できるのは PERFORCE 管理者のみとなり、変更可能なフィールドはコメントおよびユーザ名のみになります。

デフォルト・チェンジリストのサブミットの際にエラーが発生した場合、PERFORCE はサブミットしようとしたファイルを含む番号付チェンジリストを作成します。問題の修正後、-c オプションを使用して番号付チェンジリストをサブミットしなければなりません。

PERFORCE は編集目的で作業状態にしたファイルを書き込み可能にして、それらのファイルを含むチェンジリストのサブミットが成功したときに書き込み禁止にします。PERFORCE サーバによるワークスペース管理と衝突しないようにするため、ファイルの書き込み権限を手動で変更しないでください。

チェンジリストの削除

作業中チェンジリストを削除するには、まずそのチェンジリストに関連付けられているすべてのファイルおよびジョブを削除した後、p4 change -d *changenum* コマンドを発行します。関連する操作には以下のものが含まれます。

- ファイルを別のチェンジリストに移動するには、p4 reopen -c *changenum* コマンドを発行します。
- ファイルをチェンジリストから削除して変更を破棄するには、p4 revert -c *changenum* コマンドを発行します。

既にサブミットされたチェンジリストの削除は、PERFORCE 管理者のみ実行が可能です。詳しくは、*『PERFORCE システム管理者ガイド』* をご覧ください。

ファイルのリネームと移動

ファイルのリネームまたは移動を行うには、p4 delete コマンドと p4 integrate コマンドを同時に使用して、新しいファイルを作成するとともに元のファイルを削除し、リビジョン履歴を維持します。手順は以下のとおりです。

```
p4 integrate source_file target_file
p4 delete source_file
p4 submit
```

ファイルの集合をリネームするには、*source_file* 指定子と *target_file* 指定子とで一致するワイルドカードを使用します。ファイルをリネームするには、指定されたファイルに対する PERFORCE の write 権限が必要です。(PERFORCE 権限に関して詳しくは、*『PERFORCE システム管理者ガイド』* をご覧ください。)

p4 integrate を使用してファイルのリネームまたは移動を行うとき、PERFORCE サーバは削除された先行ファイルにリンクする反映記録を作成して、ファイルの履歴を保存します。(反映操作はブランチの作成および変更の伝達にも使用されます。詳しくは、68 ページの「変更の反映」をご覧ください。

チェンジリストに関する情報の表示

チェンジリストの簡潔な情報を表示するには、p4 changes コマンドを使用します。詳細な情報を表示するには、p4 describe コマンドを使用します。以下の表に有用なレポート・コマンドおよびオプションをいくつか記述します。

コマンド	説明
p4 changes	すべての作業中チェンジリストおよびサブミット済チェンジリストの一覧を表示します。チェンジリストあたり 1 行の簡単なコメントが付きます。
p4 changes -m *count*	レポートされるチェンジリストの数を、最新の count 数のチェンジリストに制限します。
p4 changes -s *status*	特定の状態 status のチェンジリストだけの一覧を表示します。例えば p4 changes -s submitted の場合は、サブミット済チェンジリストだけが表示されます。
p4 changes -u *user*	特定のユーザ user によってサブミットされたチェンジリストの一覧を表示します。
p4 changes -c *workspace*	特定のクライアント・ワークスペース workspace からサブミットされたチェンジリストの一覧を表示します。
p4 describe *changenum*	単一のチェンジリストに関するすべての情報を表示します。サブミット済チェンジリストの場合、対象ファイルの一覧および処理済みファイルの差分もレポートに含まれます。ファイルの差分を除外したいときは -s オプションを使用します。

より詳しくは、89 ページの「チェンジリストのレポート」をご覧ください。

ファイル内容の比較

PERFORCE にはテキスト・ファイルのリビジョンの*差分を取る*（比較する）ことが可能なプログラムがあります。ファイル内容を比較することにより、次のような内容を表示できます。

- ファイルを編集目的で作業状態にした後に行った変更
- 2 つのリビジョン間の差分
- 異なるブランチにあるファイル・リビジョン間の差分

ワークスペースに同期されたファイルをディポ・リビジョンと比較するには、`p4 diff` *`filename#rev`* コマンドを発行します。リビジョン指定子を省略すると、ワークスペース内のファイルは、最後に同期させたリビジョンと比較され、ファイルを同期させた後に行った変更が表示されます。

ディポには存在しワークスペースには存在しない 2 つのリビジョンを比較するには、`p4 diff2` コマンドを使用します。ファイルの集合を比較するには、`p4 diff2` コマンドの発行時にファイル名の引数にワイルドカードを指定します。

`p4 diff` コマンドではクライアント・マシン上で比較が行われますが、`p4 diff2` コマンドではサーバ・マシン上で比較が行われ、その結果がクライアント・マシンに送られます。

以下の表に比較コマンドに共通する用途をいくつか示します。

比較する	比較対象	使用するコマンド
ワークスペース・ファイル	最新リビジョン	`p4 diff` *`file`* または `p4 diff` *`file`*`#head`
ワークスペース・ファイル	リビジョン 3	`p4 diff` *`file`*`#3`
最新リビジョン	リビジョン 134	`p4 diff2` *`file file`*`#134`
チェンジリスト 32 のファイル・リビジョン	チェンジリスト 177 のファイル・リビジョン	`p4 diff2` *`file`*`@32` *`file`*`@177`
リリース 1 のすべてのファイル	リリース 2 のすべてのファイル	`p4 diff2 //depot/rel1/... //depot/rel2/...`

デフォルトでは、`p4 diff` コマンドにより PERFORCE クライアント内部の比較プログラムが起動されます。異なる比較プログラムを使用するには、`P4DIFF` 環境変数を設定して目的のプログラムのパスと実行ファイル名を指定します。外部の比較プログラムに引数を指定するには、`-d` オプションを使用します。詳しくは、*『PERFORCE コマンド・リファレンス』* を参照してください。

非接続で作業する

PERFORCE が制御しているファイルを非接続で（PERFORCE サーバにアクセスせずに）操作する必要がある場合、サーバに再度アクセスしたときにそれまで行った作業を PERFORCE サーバと一致させなければなりません。以下に示す非接続での作業方法は、クライアント・ワークスペースにあるファイルを操作する場合、またはディポを更新する前に追加、更新、削除を行ってワークスペースを更新する場合を想定しています。

非接続での作業に関するプラットホーム固有の情報は、PERFORCE Web サイトに掲載の Tech Note 2 をご覧ください。

```
http://www.perforce.com/perforce/technotes/note002.html
```

非接続で作業するには

1. p4 コマンドを発行せずに、ファイルを操作します。または、OS のコマンドを使ってファイルのパーミッションを変更してください。
2. ネットワーク接続が再確立されたら、p4 diff を使用してワークスペース内のすべての変更されたファイルを見つけます。(新しいファイルは手動で追跡する必要があります。)
3. ファイルを必要に応じて追加、編集、または削除目的の作業状態にして、結果となるチェンジリストをサブミットしてディポを更新します。

次のセクションで、より詳しく説明します。

変更されたファイルを見つける

変更されたファイルを検出するには、p4 diff コマンドを発行します。以下に示すオプションを使って、実際にファイルを PERFORCE で編集目的または削除目的の作業状態にすることなく、手動で変更または削除を行ったファイルの場所を特定することができます。

オプション	説明
p4 diff -se	編集目的の作業状態ではないが、同期後に変更されているワークスペース・ファイルをリスト表示します。これらのファイルでディポを更新するには、編集目的で作業状態にしてからサブミットします。
p4 diff -sd	削除目的の作業状態ではないが、手動でワークスペースから削除されたファイルをリスト表示します。これらのファイルの削除を反映させてディポを更新するには、削除目的で作業状態にしてからサブミットします。

変更のサブミット

非接続での作業によりクライアント・ワークスペースに加えた変更でディポを更新するには、以下の例に示すように、上述したオプションに -x を組み合わせて p4 diff を使用します。-x オプションは p4 edit コマンドに対して、引数をパイプ（またはファイル）から受け取るよう指示します。

非接続で作業した後、変更されたファイルを編集目的で作業状態にするには、次のコマンドを発行します。

```
p4 diff -se | p4 -x - edit
```

クライアント・ワークスペースから除去されたファイルをディポから削除するには、次のコマンドを発行します。

```
p4 diff -sd | p4 -x - delete
```

任意の新しいファイルを追加目的の作業状態にし、追加、更新、削除の内容を含むチェンジリストをサブミットします。

第 5 章

衝突の解決

この章では、チーム開発環境での作業方法を説明します。チーム開発環境では、同一のファイルを複数のユーザが操作するため、各ユーザの変更を調整する必要がある場合があります。

複数のユーザが同一のファイル群を操作するように設定されていると、衝突が発生することがあります。PERFORCE では、同じファイルのチーム内での同時操作が可能であり、発生する衝突を解決することができます。例えば、2 人のユーザが同じファイルを操作するか (チーム設定における大きな懸念事項)、またはファイルの最新リビジョンではなく古いリビジョンを編集する場合に衝突が発生します。

ディポ内の最新リビジョンと衝突するファイルをサブミットしようとすると、PERFORCE はその衝突を解決するよう要求します。変更を開発ブランチからリリース・ブランチにマージする作業も、ファイルの衝突解決が必要となる主なタスクの 1 つです。

衝突を防ぐため、PERFORCE ではファイルを編集するときにファイルをロックできます。しかし、ロックするとチームでの開発作業が制限されるおそれがあります。チーム開発では、ファイルをなるべく利用可能な状態にすると同時に衝突を最小限にとどめる手法を選択する必要があります。詳しくは、61 ページの「ファイルのロック」を参照してください。

P4V (PERFORCE ビジュアル・クライアント) および関連ビジュアル・マージ・ツールの P4Merge などの GUI ツールを使用してファイルの衝突解決を行うこともできます。

衝突が発生する状況

2 人のユーザが同じファイルの 2 つのバージョンを編集してサブミットすると、ファイルの衝突が発生することがあります。例えば次のような場合、衝突が発生します。

1. ブルーノが `//depot/dev/main/jam/command.c#8` を編集目的の作業状態にします。
2. 次にゲイルが、同じファイルを自分のクライアント・ワークスペースで作業状態にします。
3. ブルーノとゲイルはともに `//depot/dev/main/jam/command.c#8` を編集します。
4. ブルーノが `//depot/dev/main/jam/command.c` を含むチェンジリストをサブミットし、そのサブミットが成功します。
5. ゲイルは彼女のバージョンの `//depot/dev/main/jam/command.c` を含むチェンジリストをサブミットします。ゲイルのサブミットは失敗します。

PERFORCE がディポにゲイルのバージョンを受け入れると、彼女の変更によりブルーノの変更が上書きされてしまいます。PERFORCE サーバは、ブルーノの変更が失われるのを防ぐため、チェンジリストを拒否して衝突するファイルの解決をスケジュールします。ファイルが衝突することが事前に分かっており、ファイルの衝突解決をスケジュールしたい場合は、そのファイルを同期させます。PERFORCE はファイルの衝突を検出し、その解決をスケジュールします。

衝突解決の方法

ファイルの衝突を解決するには、p4 resolve コマンドを発行してサブミットを行うファイルのコンテンツを決定し、各ファイルに対して解決方法を選択します。衝突を解決したら、そのファイルを含むチェンジリストをサブミットします。

> **注** ファイルを編集目的で作業状態にした後、それ以降にサブミットされたリビジョンをディポから同期させる場合、ユーザによる変更がディポ・ファイルにより上書きされないよう、PERFORCE により衝突解決が要求されます。

PERFORCE は、デフォルトでは比較プログラムを使用して衝突を検出します。サード・パーティの比較プログラムを設定することもできます。詳しくは、50 ページの「ファイル内容の比較」をご覧ください。

衝突を解決して変更をサブミットするには、以下の手順を実行します。

1. ファイル(例えば、p4 sync //depot/dev/main/jam/...)を同期させます。PERFORCE は衝突を検出し、衝突しているファイルの解決をスケジュールします。
2. p4 resolve コマンドを発行して、衝突を解決します。解決オプションについて詳しくは、55 ページの「衝突解決のオプション」をご覧ください。
3. 生成されたファイルを検査します（コードをコンパイルし、稼働を確認するなどして)。
4. ファイルを含むチェンジリストをサブミットします。

> **注** マージ対象の 3 つのファイル・リビジョンのいずれかがテキストではなくバイナリである場合、3 ウェイ・マージを行うことはできません。代わりに、p4 resolve により 2 ウェイ・マージが実行されます。ファイルの衝突する 2 つのバージョンが表示されるので、どちらかを選択するか、いずれか 1 つをワークスペースで編集してからチェンジリストをサブミットします。

yours、theirs、base、merge の各ファイル

p4 resolve コマンドではマージ処理において次の用語が使用されます。

ファイル・リビジョン	説明
yours	クライアント・ワークスペースにある、変更を含むファイルのリビジョン。
theirs	他のユーザにより編集され yours と衝突している、ディポ内のリビジョン。(通常は最新リビジョンであるが、p4 sync を使って別のリビジョンによる衝突解決のスケジュールが可能。)
base	*yours* と theirs を編集する元となったディポ内のファイル・リビジョン（直近の共通祖先となるファイル)。
merge	PERFORCE により *theirs*、*yours*、*base* から生成されるファイル。
result	衝突解決プロセスの最終結果となるファイル。

衝突解決のオプション

衝突を解決する方法を指定するには、p4 resolve コマンドを発行します。それにより衝突解決がスケジュールされたファイルごとに、ダイアログが表示されます。ダイアログには、変更したファイルと衝突するリビジョンとの相違点が記述されます。例を示します。

```
p4 resolve //depot/dev/main/jam/command.c
c:¥bruno_ws¥dev¥main¥jam¥command.c - merging //depot/dev/main/jam/command.c#9

Diff chunks: 4 yours + 2 theirs + 1 both + 1 conflicting
Accept(a) Edit(e) Diff(d) Merge (m) Skip(s) Help(?) e:
```

各ファイルの組の相違点が p4 resolve により要約されます。行のまとまり（チャンク）は yours、theirs、base ファイルでそれぞれ異なる場合があります。チャンクの形態を以下に示します。

- **差分** : *yours*、*theirs*、*base* の 3 つのファイルのうち 2 つが異なる場合
- **衝突** : 3 つのファイルすべてが異なる場合

上記の例において、

- 4 つのチャンクが *theirs* と *base* とで同一であるが、*yours* では異なる
- 2 つのチャンクが *yours* と *base* とで同一であるが、*theirs* では異なる
- 1 つのチャンクが *yours* と *theirs* とで同様に変更されている
- 1 つのチャンクが *yours* 、*theirs*、*base* で異なっている

PERFORCE が推奨する選択がコマンドラインの終わりに表示されます。ENTER キーを押すか、Accept を選択すると、推奨された選択が実行されます。

以下に示す 3 つの基本的な方法により衝突解決が可能です。

- ファイルを変更せずに受け入れます。（56 ページの「yours、theirs、または merge の承諾」をご覧ください。）
- merge ファイルをテキスト・エディタで編集します。（57 ページの「merge ファイルの編集」をご覧ください。）
- マージ・プログラムを使用して、選択を行いながら変更をマージします。（57 ページの「衝突解決のためのマージ」をご覧ください。）

上記のオプションは対話式オプションです。受け入れたいファイルが分かっている場合は、p4 resolve コマンドラインで解決オプションを指定することもできます。詳しくは、60 ページの「衝突解決のためのコマンドライン・オプション」をご覧ください。

解決済みであるがサブミットされていないファイルを再度解決するには、p4 resolve コマンドの発行時に -f オプションを指定します。ファイルのサブミット後は、再度解決を行うことはできません。

次のセクションでは、解決オプションについてさらに詳しく説明します。

yours、theirs、または merge の承諾

ファイルを変更せずに受け入れるには、以下のオプションの 1 つを指定します。

オプション	説明	意味
a	推奨されるファイルを承諾	• *theirs* が *base* と同じである場合、*yours* を受け入れます。 • *yours* が *base* と同じである場合、*theirs* を受け入れます。 • *yours* と *theirs* が *base* と異なり、*yours* と *theirs* に衝突がない場合、*merge* を受け入れます。 • それ以外の場合 *yours* と *theirs* に衝突があるため、このファイルをスキップします。
ae	編集結果 (edit) を承諾	(p4 resolve ダイアログから e を選択して) *merge* ファイルが編集されていれば、編集後のバージョンをクライアント・ワークスペースに受け入れます。クライアント・ワークスペース内のバージョンは上書きされます。
am	*merge* を承諾	*merge* を衝突解決後のリビジョンとしてクライアント・ワークスペースに受け入れます。クライアント・ワークスペース内のバージョンは上書きされます。
at	*theirs* を承諾	*theirs*を衝突解決後のリビジョンとしてクライアント・ワークスペースに受け入れます。クライアント・ワークスペース内のバージョンは上書きされます。
ay	*yours* を承諾	*yours* を衝突解決後のリビジョンとしてクライアント・ワークスペースに受け入れます。*theirs* に加えられた変更は無視されます。

yours、*theirs*、*edit*、または *merge* を受け入れると変更が上書きされるため、生成されたマージ・ファイルがディポにサブミットしたい内容と正確に一致しない場合があります。必要な変更のみを正しくサブミットするには、マージ・プログラムを使用するか、マージ・ファイルを編集してください。

merge ファイルの編集

マージ・ファイルを編集してファイルを解決するには、e オプションを選択します。PERFORCE はデフォルトのテキスト・エディタを起動し、マージ・ファイルを表示します。マージ・ファイルでは、差分および衝突が次のフォーマットで表示されます。

```
>>>> ORIGINAL file#n
(元のバージョンからのテキスト)
==== THEIR file#m
(their ファイルからのテキスト)
==== YOURS file
(your ファイルからのテキスト)
<<<<
```

衝突と差分のある箇所を見つけるには、差分マーカ「>>>>」を探し、テキストの該当部分を編集します。*theirs* の変更箇所を調べ、自分が行った変更と矛盾しないことを確認します。保存する前に必ず、すべての衝突マーカを削除してください。必要な変更を行った後、ファイルを保存します。p4 resolve プロンプトにおいて、ay を選択します。

デフォルトでは、*yours* ファイルと *theirs* ファイルとの衝突のみがマークされます。差分マーカがすべての差分を生成するようにするには、p4 resolve コマンドの発行時に -v オプションを指定します。

衝突解決のためのマージ

マージ・プログラムにより yours、theirs、base の各ファイル間の差分が表示されるため、そこから変更の選択や編集を行って必要な結果ファイルを作成することができます。マージ・プログラムを構成するには、P4MERGE に対象のプログラムを設定します。解決処理中にマージ・プログラムを使用するには、m オプションを選択します。特定のマージ・プログラムの使用に関して詳しくは、オンライン・ヘルプを参照してください。

マージを行ったら、結果を保存してマージ・プログラムを終了します。p4 resolve プロンプトにおいて、am を選択します。

解決オプションの全リスト

p4 resolve コマンドには、次のオプションがあります。

オプション	動作	説明
?	Help	p4 resolve のヘルプを表示します。
a	自動的に承諾	自動的に選択されたファイルを受け入れます。 ・ *theirs* が *base* と同じである場合、*yours* を受け入れます。 ・ *yours* が *base* と同じである場合、*theirs* を受け入れます。 ・ *yours* と *theirs* が *base* と異なり、*yours* と *theirs* に衝突がない場合、*merge* を受け入れます。 ・ それ以外の場合 yours と *theirs* に衝突があるため、このファイルをスキップします。
ae	編集結果 (edit) を承諾	(p4 resolve ダイアログから e を選択して) *merge* ファイルが編集されていれば、編集後のバージョンをクライアント・ワークスペースに受け入れます。クライアント・ワークスペース内のバージョンは上書きされます。
am	*merge* を承諾	*merge* を衝突解決後のリビジョンとしてクライアント・ワークスペースに受け入れます。クライアント・ワークスペース内のバージョンは上書きされます。
at	*theirs* を承諾	*theirs*を衝突解決後のリビジョンとしてクライアント・ワークスペースに受け入れます。クライアント・ワークスペース内のバージョンは上書きされます。
ay	*yours* を承諾	*yours* を衝突解決後のリビジョンとしてクライアント・ワークスペースに受け入れます。*theirs* に加えられた変更は無視されます。
d	差分	*merge* と *yours* との差分を表示します。
dm	*merge* 差分	*merge* と *base* との差分を表示します。
dt	*theirs* 差分	*theirs* と *base* との差分を表示します。
dy	*yours* 差分	*yours* と *base* との差分を表示します。
e	*merge* 編集	PERFORCE によって生成された予備マージ・ファイルを編集します。
et	*theirs* 編集	クライアント・リビジョンが衝突するディポ内のリビジョン (通常は最新リビジョン) を編集します。この編集は読み取り専用です。
ey	*yours* 編集	現在ワークスペースにあるファイル・リビジョンを編集します。
m	マージ	コマンド P4MERGE *base theirs yours merge* を実行します。このオプションを実行するには、環境変数 P4MERGE にサード・パーティのプログラムの名前を設定しなければなりません。このプログラムでは最初の 3 つのファイルをマージし、その結果を 4 番目のファイルに書き込むものとします。
s	スキップ	このファイルをスキップし、衝突解決がスケジュールされたままにします。

注 マージ・ファイルは PERFORCE サーバにより生成されますが、dy、dt、dm、d の各オプションで表示される差分は、クライアント・マシンの比較プログラムにより生成されます。衝突解決で d オプションを選択したときに他の比較プログラムが実行されるように設定するには、P4DIFF 環境変数を設定します。詳しくは、50 ページの「ファイル内容の比較」をご覧ください。

例 : *ファイルの衝突を解決する*

Jam readme ファイルに対してブルーノが行った作業とアールが行った作業との衝突を解決するため、ブルーノが`p4 resolve //depot/dev/main/jam/README` *と入力すると、以下のように表示されました。*

```
Diff chunks: 0 yours + 0 theirs + 0 both + 1 conflicting
Accept(a) Edit(e) Diff(d) Merge (m) Skip(s) Help(?) e: e
```

ブルーノは、彼とアールがファイルに対して衝突する変更を加えたことを確認しました。そこで彼はマージ・ファイルを編集するために `e` *を入力して、差分マーカ 「>>>>」を探します。すると以下のテキストが表示されます。*

```
Jam/MR (formerly "jam - make(1) redux")
/+¥
>>>> ORIGINAL README#26
    +¥Copyright 1993, 1997 Christopher Seiwald.
==== THEIRS README#27
    +¥Copyright 1993, 1997, 2004 Christopher Seiwald.
==== YOURS README
    +¥Copyright 1993, 1997, 2005 Christopher Seiwald.
<<<<
¥+/
```

ブルーノとアールは異なる著作権登録年を記述して更新しました。ブルーノはマージ・ファイルを編集してヘッダーを正しく修正し、エディタを終了して、`am` *と入力します。編集されたマージ・ファイルがクライアント・ワークスペースに書きこまれたので、彼は次のファイルの衝突解決に進みます。*

衝突解決プロセスにおいてファイルの 1 バージョンが受け入れられると、ワークスペース内のファイルは上書きされるので、新しいクライアント・ファイルをディポにサブミットする必要があります。ファイルの衝突解決が完了してからサブミットするまでの間に、そのファイルの新しいバージョンがサブミットされると、新しい衝突が発生する場合があります。衝突解決を行う前にファイルをロックすることにより、この問題を回避することができます。詳しくは、61 ページの「ファイルのロック」をご覧ください。

衝突解決のためのコマンドライン・オプション

p4 resolve オプションに次のオプションをつけると、衝突解決を対話形式ではなく直接実行できます。p4 resolve コマンドでこれらのオプションの 1 つを指定すると、ファイルは以下の表に示すように解決されます。

オプション	説明
-a	自動的に選択されたファイルを受け入れます。
-ay	*yours* を受け入れます。
-at	*theirs* を受け入れます。クライアント・ワークスペース内のファイル・リビジョンがディポからの最新リビジョンで上書きされ、変更を元に戻すことができないため、このオプションを使用するときは注意が必要です。
-am	以下のロジックに従って、推奨されるファイル・リビジョンを受け入れます。 ・ *theirs* が *base* と同一であれば、*yours* を受け入れます。 ・ *yours* が *base* と同一であれば、*theirs* を受け入れます。 ・ *yours* と *theirs* が *base* と異なり、*yours* と *theirs* との間で衝突がなければ、*merge* ファイルを受け入れます。 ・ それ以外の場合は、*yours* と *theirs* との間で衝突があるので、このファイルは解決されずにスキップされます。
-af	衝突箇所が残っていても、推奨されるファイル・リビジョンを受け入れます。このオプションを使用する場合、結果のファイルをワークスペースで編集し、すべての差分マーカを除去する必要があります。
-as	以下のロジックに従って、推奨されるファイル・リビジョンを受け入れます。 ・ *theirs* が *base* と同一であれば、*yours* を受け入れます。 ・ *yours* が *base* と同一であれば、*theirs* を受け入れます。 ・ それ以外の場合は、このファイルはスキップされます。

例 : *衝突するファイルの特定リビジョンを自動的に受け入れる*

ブルーノは、/doc にある文書ファイルを編集しており、そのいくつかが衝突解決を必要とすることを知っています。ブルーノは、p4 sync doc/.guide と入力します。ディポ内のファイルと衝突するこれらのファイルすべての衝突解決がスケジュールされます。*

ブルーノはさらに p4 resolve -am と入力します。スケジュールされた衝突解決に関するマージ・ファイルがすべて生成され、そのうち行セット衝突を含まないマージ・ファイルが彼のクライアント・ワークスペースに書き込まれます。ブルーノはなお、他の衝突するファイルについて手動で解決を行う必要がありますが、その作業量はかなり軽減されました。

衝突解決のレポート・コマンド

次に示すレポート・コマンドは、ファイルの衝突解決の際に役立ちます。

コマンド	意味
`p4 diff [filenames]`	ワークスペース内にあるファイル・リビジョンと最後に同期したリビジョンとを比較し、加えられた変更を表示します。
`p4 diff2 file1 file2`	2 つのディポ・ファイルを比較します。指定ファイルには任意のファイル・リビジョンを選択でき、異なるファイルのリビジョンであっても構いません。 ディポ・ファイルを比較する場合、PERFORCE サーバは独自の比較プログラムを使用します。P4DIFF 環境変数で設定された比較プログラムは使用されません。
`p4 sync -n [filenames]`	指定された同期の結果をプレビューし、衝突していて解決が必要なファイルをリスト表示します。
`p4 resolved`	衝突解決された後にまだサブミットされていないファイルをレポートします。

ファイルのロック

ファイルを作業状態にした後、そのファイルをロックして、自分がサブミットする前に他のユーザがサブミットできないようにすることができます。ファイルをロックすると衝突を防止することができますが、ファイルのロック中は他のチーム・メンバーがそのファイルを操作できない可能性があります。

ファイルをロックして、多重衝突解決を防止する

ファイル・ロックをしない限り、衝突解決プロセスが終了する保証はありません。次のシナリオは、この問題を具体的に表現しています。

1. ブルーノがファイルを編集目的で作業状態にする。
2. ゲイルが同じファイルを、自分のクライアントにおいて編集目的で作業状態にする。
3. ブルーノとゲイルは、ファイルの各自のクライアント・ワークスペース・バージョンを編集する。
4. ブルーノは当該ファイルを含むチェンジリストをサブミットし、このサブミットが成功する。
5. ゲイルも自分のファイル・バージョンを含むチェンジリストをサブミットするが、そのファイルがディポの新しいファイルと衝突するため、サブミットは失敗する。
6. ゲイルが衝突解決を始める。
7. ブルーノが同じファイルの新しいバージョンを編集し、サブミットする。
8. ゲイルが衝突解決を終え、サブミットしようとする。サブミットは失敗し、今度はブルーノの最新ファイルとマージしなければならない。

   ＜等々＞

このような問題を回避するため、ファイルをロックすることができます。手順は次のとおりです。

1. 衝突解決をスケジュールする前に、ファイルをロックします。
2. ファイルを同期させます（衝突解決をスケジュールするため）。
3. ファイルの衝突を解決します。
4. ファイルをサブミットします。
5. チェンジリストのサブミットが成功すると、PERFORCE は自動的にファイルのロックを解除します。

UNIX において作業状態にありロックされているファイルをリストするには、次のコマンドを発行します。

```
p4 opened | grep "*locked*"
```

多重チェックアウトを防止する

一度に 1 人のユーザだけがファイルを操作できるようにするには、ファイルタイプ修飾子 +l（排他的作業状態）を使用します。例を示します。

```
p4 reopen -t binary+l file
```

排他的ロックにより同時並行開発ができなくなりますが、マージや衝突解決をする意味がない一部のファイルタイプ（通常はバイナリ・ファイル）については、複数のユーザが同時にファイルの操作を行わないようにすることで衝突を回避できます。

PERFORCE 管理者は、p4 typemap コマンドを使用して、特定のタイプ（例えば、//depot/.../*.gif としてすべての .gif ファイルを指定）を持つすべてのファイルについて、一度に 1 人のユーザだけが作業状態にできるように設定することができます。詳しくは、*『PERFORCE コマンド・リファレンス』* をご覧ください。

p4 lock と +l との違いは次のとおりです。p4 lock の場合は誰でもファイルを編集目的で作業状態にすることができますが、ファイルをロックした当人のみがそのファイルをサブミットできます。これに対し、ファイルタイプ +l のファイルは、一度に複数のユーザが作業状態にすることはできません。

第 6 章

コードラインとブランチ機能

この章では、ディポ内のファイルの集合を保守するために必要なタスクについて説明します。具体的には、次のような問題に対応します。

- ディポ・ディレクトリ構造およびリポジトリの最適な編成方法
- コードラインおよびプロジェクト・ディレクトリにおける、ファイルとファイル変更の移動
- ラベルまたはチェンジリストを使用した、ファイル・セットの特定

この章ではソフトウェア・コードベースの保守に焦点をあてていますが、記述するタスクの多くは、Web サイトなど他のファイル集合の管理にも関連します。最善の実装方法については、PERFORCE ウェブサイトにある報告書をご覧ください。

基本用語

以降のセクションを理解しやすくするため、PERFORCE での関連用語の定義を次に示します。

用語	定義
ブランチ	(*名詞*) 新規追加ではなくコピーにより作成された、関連するファイルのセット。関連するファイルの集合は、多くの場合*コードライン*と呼ばれます。 (*動詞*) ブランチを作成すること。
反映	ファイルの祖先を保存して既存のファイルから新しいファイルを作成すること（ブランチ作成）、または 1 つのファイル・セットから別のファイル・セットに変更を伝達すること（マージ処理）。
マージ	通常 P4Merge などのマージ・ツールを使用して、2 つの衝突しているファイル・リビジョンの内容を単一のファイルに結合するプロセス。
衝突解決	2 つのファイル・リビジョン間の差分を調整するために使用するプロセス。衝突解決の方法として、サブミット対象のファイルを選択するか、衝突しているファイルの内容をマージするかを選択できます。

ディポの編成

ディポは、最上位のディレクトリと考えられます。次に示す要素を考慮してディポの編成方法を決定してください。

- **内容のタイプ** : 自分のプロジェクトおよびそれらの相互関係の性質に基づいて、ディポまたは本流となるディレクトリを作成します（例えば、別々のスケジュールで開発された複数のコンポーネントを含むアプリケーション）。
- **リリース要件** : 1 つのプロジェクト内に、リリースごとにブランチを作成し、ブランチ間の変更を反映させて、機能およびバグ修正の導入を制御します。
- **ビルドの管理** : ラベルとチェンジリストを使用して、ビルドされたファイル・リビジョンを制御します。クライアント仕様およびビューを使用して、ビルド範囲が明確になるようにします。

基本的かつ論理的にディポを編成する方法は、プロジェクトごとにサブディレクトリ（コードライン）を作成することです。例えば、自分の会社が Jam プロジェクトに関わっている場合、1 つのコードラインを現在開発中のリリース専用にし、もう 1 つのコードラインを既にリリース済のソフトウェア専用にし、さらにもう 1 つを会社の Web サイト専用にすることが考えられます。開発者は各自のクライアント・ビューを変更して、ファイルを自分のプロジェクトにマッピングでき、関係のない他のプロジェクトを除外できます。例えば、アールが Web サイトの保守を担当している場合、彼のクライアント・ビューは次のようになります。

```
//depot/www/dev/... //earl-web-catalpa/www/development/...
//depot/www/review/... //earl-web-catalpa/www/review/...
//depot/www/live/... //earl-web-catalpa/www/live/...
```

そして、Jam に携わっているゲイルは自分のクライアント・ビューを次のように設定します。

```
//depot/dev/main/jam/... //gale-jam-oak/jam/...
```

プロジェクトまたはコードラインの用途に従ってディポを編成することができます。例えば、プロジェクトに従ってディポを編成するには、次のような構造を使用できます。

```
//depot/project1/main/
//depot/project1/release 1.0/
//depot/project1/release 1.1/
```

一方、各コードラインの用途に従ってディポを編成するには、次のような構造を使用できます。

```
//depot/main/project1/
//depot/main/project2/
//depot/release1.0/project1/
//depot/release1.0/project2/
//depot/release2.0/project1/
//depot/release2.0/project2/
```

他には、各プロジェクトに対して 1 つずつディポを作成する方法があります。ブランチ作成および反映を可能な限り単純化できる構造を選択し、操作の履歴が理解しやすいようにしてください。

ブランチ機能

ブランチ機能は、関連するファイル・セット間の関係を保守する方法です。ブランチは、それ自身の祖先や子孫から独立して発展させることができ、必要に応じて 1 つのブランチから別のブランチに変更を伝達（*反映*）することができます。PERFORCE の *インター・ファイル・ブランチ* ™ 機能により、サーバ・リソースの消費を最小限に留めながら、ファイルとその祖先との関係を保持します。

ブランチを作成するには、`p4 integrate` コマンドを使用します。`p4 integrate` コマンドは既存のファイル・セット間での変更の伝達にも使用されます。変更の反映に関して詳しくは、68 ページの「変更の反映」を参照してください。

ブランチを作成するタイミング

2 つのファイル・セットにおいて、サブミットの方針が異なる場合、または別々に展開させる必要がある場合には、ブランチを作成します。例を示します。

- *問題*: 開発グループのメンバーは、コードのコンパイルの有無に関係なく、コードが変更されるたびにそのコードをディポにサブミットしたい。しかし、リリース・エンジニアは、デバッグ、検証および承認が終わるまで、コードをサブミットしたくない。

  解決策: 開発コードラインをブランチして、リリース・ブランチを作成します。開発コードラインで準備ができたら、リリース・コードラインに反映します。パッチやバグ修正がリリース・コードで行われ、開発コードに逆に反映されます。

- *問題*: ある会社で、新しいマルチ・プラットフォーム・プリンタ用のドライバを書いています。彼らは、UNIX デバイス・ドライバを書き終え、今後は UNIX コードを基にして、Macintosh ドライバを書く作業を始めようとしています。

  解決策: 既存の UNIX コードから Macintosh ブランチを作成します。これらのコードラインは別々に展開できます。一方のコードラインでバグが見つかった場合、バグ修正を他方のコードラインに伝達することができます。

基本戦略の 1 つは、`//depot/main/` でコードを開発する一方で、リリース用のブランチを作成することです（例えば、`//depot/rel1.1/`）。リリース固有のバグ修正をリリース・ブランチで行い、必要に応じて、`//depot/main/` コードラインに逆に反映させます。

ブランチの作成

ブランチを作成するには、p4 integrate コマンドを使用します。ブランチを作成すると、PERFORCE サーバはブランチされたファイルとそれらの祖先との関係を記録します。

ブランチは、ファイル指定またはブランチ仕様を使用して作成できます。単純なブランチの場合は、ファイル指定を使用します。複雑なファイル・セット構成に基づくブランチの場合、またはブランチの定義方法を記録しておきたい場合は、ブランチ仕様を使用します。ブランチ仕様は、その後の反映処理にも使用できます。またブランチ仕様は、コードラインの方針の記録としても役立てることができます。

ブランチ仕様を使用する

反映元ファイル・セットを反映先ファイルにマッピングするために、*ブランチ仕様*を作成し、これを p4 integrate の引数として使用することができます。ブランチ仕様を作成するには、p4 branch *branchname* コマンドを発行し、対象とするマッピングを View: フィールドに指定します。このとき反映元のファイルを左側、反映先のファイルを右側に示します。反映先ファイルとディレクトリがクライアント・ビューにあることを確認してください。ブランチ仕様を作成または変更しても、ディポやクライアント・ワークスペースにあるファイルには一切影響しません。ブランチ仕様は、反映元ファイルを反映先ファイルにマッピングするだけです。

ブランチ仕様を使用してブランチを作成するには、p4 integrate -b *branchname* コマンドを発行します。その後、p4 submit を使用して反映先のファイルをディポにサブミットします。

ブランチ仕様には、クライアント・ビューと同様に、複数のマッピングや排他的なマッピングを含めることができます。例えば、次に示すブランチ仕様により、テスト用スクリプトを除く Jam 1.0 ソースコードがメインのコードラインからブランチ作成されます。

```
Branch:   jamgraph-1.0-dev2release

View:
    //depot/dev/main/jamgraph/... //depot/release/jamgraph/1.0/...
    -//depot/dev/main/jamgraph/test/... //depot/release/jamgraph/1.0/test/...
    //depot/dev/main/bin/glut32.dll //depot/release/jamgraph/1.0/bin/glut32.dll
```

上記のブランチ仕様を使用してブランチを作成するには、次のコマンドを発行します。

```
p4 integrate -b jamgraph-1.0-dev2release
```

ブランチ仕様を削除するには、p4 branch -d *branchname* コマンドを発行します。ブランチ仕様を削除しても、既存のファイルやブランチには影響しません。

ワークスペース・ビューと同様に、ブランチ・ビューのファイル名またはパスに空白が含まれている場合は、必ずパスを引用符で囲んでください。

```
//depot/dev/main/jamgraph/... "//depot/release/Jamgraph 1.0/..." f
```

ファイル指定を使用する

ファイル指定を使用してブランチを作成するには、反映元ファイルと反映先ファイルを指定して p4 integrate コマンドを発行します。反映先ファイルはクライアント・ビューに存在する必要があります。反映元ファイルがクライアント・ビューに存在しない場合、ディポ構文を使用して指定してください。

ファイル指定を使用してブランチを作成するには、以下の手順で操作します。

1. ブランチされるファイルを、ディポ内およびクライアント・ワークスペース内のどこに置きたいかを決めます。対応するマッピングを自分のクライアント・ビューに追加します。
2. p4 integrate *source_files target_files* コマンドを発行します。
3. ブランチされたファイルを含むチェンジリストをサブミットします。反映先ファイルを含むブランチがディポに作成されます。

例 : *ファイル指定を使用してブランチを作成する*

Jam のバージョン 2.2 がちょうどリリースされたばかりで、バージョン 3.0 の作業が始まろうとしています。バージョン 2.2 を保守作業のために //depot/release/jam/2.2/... *にブランチする必要があります。*

ブルーノは p4 client *を使って、次のマッピングを自分のクライアント・ビューに追加します。*

```
//depot/release/jam/2.2/... //bruno_ws/release/jam/2.2/...
```

ブルーノは次のコマンドを発行してブランチを作成します。

```
p4 integrate //depot/dev/main/jam/...
//bruno_ws/release/jam/2.2/...
```

最後に、ブルーノは p4 submit *コマンドを発行します。これで新しくブランチされたファイルがディポに追加されます。*

変更の反映

ブランチを作成したら、一方から他方に変更を伝達する必要があるかもしれません。例えば、リリース・ブランチでバグを修正したら、その修正をメインのコードラインに取り入れたいと考えるでしょう。選択した変更をブランチされたファイル間で伝達するには、次のとおり `p4 integrate` コマンドを使用します。

1. `p4 integrate` ファイルの衝突解決をスケジュールします。
2. `p4 resolve` コマンドを発行し、変更を反映元ファイルから反映先ファイルにコピーします。

   個々の変更をコピーするには、マージ・ファイルを編集するか、マージ・プログラムを使用します。変更がクライアント・ワークスペース内の反映先ファイルに加えられます。
3. 衝突解決されたファイルを含むチェンジリストをサブミットします。

例 : *ブランチ・ファイル間で変更を伝達する*

ブルーノは、Jam プロジェクトのリリース 2.2 ブランチにおけるバグを修正したので、バグ修正をメインのコードラインに反映させる必要があります。ブルーノは自分のホーム・ディレクトリから次のように入力します。

```
p4 integrate //depot/release/jam/2.2/src/Jambase
//depot/dev/main/jam/Jambase
```

すると次のようなメッセージが表示されます。

```
    //depot/dev/main/jam/Jambase#134 - integrate from
    //depot/release/jam/2.2/src/Jambase#9
```

ファイルに対して、衝突解決がスケジュールされました。ブルーノが `p4 resolve` *と入力すると、次の標準マージ・ダイアログが画面に表示されます。*

```
    //depot/dev/main/jam/Jambase - merging
    //depot/release/jam/2.2/src/Jambase#9

    Diff chunks: 0 yours + 1 theirs + 0 both + 0 conflicting
    Accept(a) Edit(e) Diff(d) Merge (m) Skip(s) Help(?) [at]:
```

ブルーノはファイル衝突を解決します。これを行うと、結果ファイルによってブルーノのワークスペースのファイルは上書きされます。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。

`p4 integrate` コマンドを実行するには、反映先ファイルの PERFORCE 書き込み権限(`write`)と、反映元ファイルの読み取り権限（`read`）が必要です。(PERFORCE の権限については、*『システム管理者ガイド』*を参照してください。)

デフォルトでは、`p4 integrate` によりクライアント・ワークスペースに新規作成されたファイルは、最初のサブミットに先立って編集することはできません。新たに反映されたファイルをサブミット前に編集するには、衝突を解決した後、`p4 edit` コマンドを発行します。

反映対象のリビジョンの範囲に削除済リビジョンが含まれている場合（例えば、ファイルがディポから削除された後に再度追加された場合）、削除済リビジョンの反映方法を、`-d` オプションまたは `-D` オプションにより指定できます。詳しくは、*『PERFORCE コマンド・リファレンス』*を参照してください。

ブランチ仕様による反映

1 つのファイル・セットおよびディレクトリから変更を反映させるには、p4 integrate コマンドの発行時にブランチ仕様を使用することができます。ブランチ仕様を使用した p4 integrate コマンドの基本構文は以下のとおりです。

```
p4 integrate -b branchname [tofiles]
```

反映先ファイルは、ブランチ・ビューとクライアント・ビューの両方でマッピングされている必要があります。反映元ファイルは、クライアント・ビュー内にある必要はありません。*tofiles* 引数を省略すると、ブランチ内にあるすべてのファイルが操作対象となります。

ブランチ仕様を使用して反映操作を行う場合、逆の方向に反映するには -r オプションを指定します。このオプションにより、2 つのブランチ間でどちらかの方向に反映操作を行うことができます。それぞれの方向についてブランチ仕様を作成する必要はありません。

例 : *ブランチ内にある単一のファイルに変更を反映する*

メインの Jam コードラインに機能が 1 つ追加されました。ブルーノはこの機能をリリース 1.0 に伝達したいと考えます。そこで、彼は次のように入力します。

```
p4 integrate -b jamgraph-1.0-dev2release *.c
```

すると、次のテキストが表示されます。

```
//depot/release/jam/1.0/src/command.c#10 - integrate from
//depot/dev/main/jam/command.c#97
```

ファイルに対する衝突解決がスケジュールされました。ブルーノが p4 resolve *と入力すると、次の標準マージ・ダイアログが画面に表示されます。*

```
//depot/release/jam/1.0/src/command.c - merging
//depot/dev/main/jam/command.c#97

Diff chunks: 0 yours + 1 theirs + 0 both + 0 conflicting
Accept(a) Edit(e) Diff(d) Merge (m) Skip(s) Help(?) [at]:
```

彼はファイル衝突を解決します。これを行うと、ブランチされたクライアント・ワークスペース内のファイルが結果ファイルによって上書きされます。これをさらにディポにサブミットしなければなりません。

関連性のないファイル間での反映

反映先ファイルが反映元ファイルからブランチされていない場合、*base*（共通の祖先）リビジョンが存在しません。関連性のないファイル間で反映操作を行うには、-i オプションを指定します。PERFORCE は反映元ファイルの最初の（最も最近追加された）リビジョンをベース・リビジョンとして使用します。この操作を、*ベースのない*マージと呼びます。

特定のファイル・リビジョンを反映させる

デフォルトでは、integrate コマンドは最後に反映された反映元リビジョン以降のすべてのリビジョンを、反映先ファイルに反映します。反映対象のリビジョン範囲は、反映操作時に指定することができます。それにより編集中にマージ・ファイルから不要なリビジョンを手動で削除する必要がなくなります。この場合、*base* ファイルは共通の祖先となります。

例 : *特定のファイル・リビジョンを反映させる*

ブルーノはメインのコードラインで、//depot/dev/main/jam/scan.c *に対して 2 つのバグを修正しました。アールはその変更をリリース 1.0 ブランチに反映させたいと考えています。*scan.c *には種々の修正がサブミットされ、既に 20 リビジョンまで進んでいますが、アールは自分が必要としているバグ修正がチェンジリスト 30 でサブミットされたファイル・リビジョンで適用されたことを知っています。そこでアールは次のように入力します。*

```
p4 integrate -b jamgraph-1.0-dev2release
//depot/release/jam/1.0/scan.c@30,@30
```

反映先ファイル(//depot/release/jam/1.0/scan.c*) が引数として与えられていますが、ファイル・リビジョンの指定は反映元に適用されます。アールが* p4 resolve *を実行すると、ブルーノのファイルのうち、チェンジリスト 30 でサブミットされたファイル・リビジョンだけが衝突解決のためにスケジュールされます。すなわち、アールはブルーノがチェンジリスト 30 で* scan.c *に加えた変更のみを見ることになります。チェンジリスト 29 においてディポ内に存在したファイル・リビジョンが base ファイルとして使用されます。*

ファイルを再反映させ、衝突を再解決する

反映元ファイルのあるリビジョンが反映先ファイルに反映されると、そのリビジョンは通常、同じ反映先ファイルに対する次回以降の反映時にはスキップされます。既に反映されたファイルを強制的に反映させたいときは、p4 integrate コマンドに -f オプションを指定します。

同様に、衝突解決されたが（まだ）サブミットされていない反映先ファイルの場合は、p4 resolve コマンドに -f オプションを指定することにより衝突の再解決ができます。衝突の再解決を行った場合、*yours* のファイルが新しいクライアント・ファイル、すなわち、前の解決結果となります。

反映に関するレポート・コマンド

次に示すレポート・コマンドにより、ブランチおよび統合の操作対象となるファイルの状態に関して有用な情報が得られます。プレビュー・オプション（-n）は、レポート目的で使用できます。

表示情報	使用コマンド
反映の結果をプレビュー	p4 integrate -n [*filepatterns*]
衝突解決がスケジュールされたファイル	p4 resolve -n [*filepatterns*]
衝突解決されたが、まだサブミットされていないファイル	p4 resolved
ブランチ仕様のリスト	p4 branches
指定されたファイルの反映履歴	p4 integrated *filepatterns*
指定されたファイルのリビジョン履歴（指定されたファイルのブランチ元のファイルの反映履歴を含む）	p4 filelog -i [*filepatterns*]

ラベルの使用

PERFORCE の *ラベル*とは、タグ付けされた一連のファイル・リビジョンを意味します。例えば、特定のリリースを構成するファイル・リビジョンにラベル release2.0.1 をタグ付けします。一般的には、次の場合にラベルを使用できます。

- ソフトウェアの特定のリリースに含まれている全ファイル・リビジョンを追跡するとき
- 特定のファイル・リビジョンの集合（標準の構成ファイル）を他のユーザに渡したいとき
- 新規のビルド用ワークスペースを設定するとき
- 開発のためにブランチするファイル・リビジョンの集合を指定するとき
- 複数のリビジョンを 1 つのグループとしてクライアント・ワークスペースと同期させるとき

ラベルとチェンジリスト番号は両方とも、特定のファイル・リビジョンの集合を参照しますが、以下に示す相違点があります。

- ラベルにより任意のファイル・リビジョンの集合を参照できます。チェンジリスト番号は、チェンジリストがサブミットされた時点にディポ内にあるすべてのファイルの内容を参照します。ある異なる時点でのファイル・リビジョンのグループを参照する必要がある場合は、ラベルを使用します。ある一定の時点で、ファイルがその用途で一致している場合は、チェンジリスト番号を使用します。
- ラベルの内容を変更することができます。サブミット済チェンジリストの内容を変更することはできません。
- ラベルには独自に名前を付けることができます。チェンジリスト番号は PERFORCE サーバによって割り当てられます。

チェンジリストは伝統的にラベルを使用する多くのアプリケーションに適しています。ラベルと異なり、チェンジリストは特定の時点におけるファイル・セットの状態を表します。ラベルが必要であると見なす前に、単にチェンジリスト番号を参照することで要件が満たせないかどうか考えてみてください。

ラベルでファイルにタグ付けする

ファイル・リビジョンの集合に（既にタグ付けされたリビジョンに加えて）タグを付けるには、p4 tag を使用して、ラベル名および対象とするファイル・リビジョンを指定します。

例えば、//depot/release/jam/2.1/src/ の下にあるファイルの最新リビジョンに jam-2.1.0 というラベルでタグを付けるには、次のコマンドを発行します。

```
p4 tag -l jam-2.1.0 //depot/release/jam/2.1/src/...
```

最新リビジョン以外のリビジョンにタグを付けるには、次のファイル・パターンでチェンジリスト番号を指定します。

```
p4 tag -l jam-2.1.0 //depot/release/jam/2.1/src/...@1234
```

指定したラベルをタグ付けできるのは、指定ファイルの 1 つのリビジョンだけですが、同一のファイル・リビジョンに複数のラベルをタグ付けすることができます。

ファイルからタグを取り外す

ファイル・リビジョンからタグを取り外すには、次のように実行します。

```
p4 tag -d -l labelname filepattern
```

このコマンドは、指定されたラベルとそれによってタグ付けされたファイル・リビジョンとの間の関連付けを削除します。例えば、//depot/release/jam/2.1/src/... の下にあるすべてのリビジョンが jam-2.1.0 でタグ付けされている場合、次のようにしてヘッダファイルのタグのみを取り外すことができます。

```
p4 tag -d -l jam-2.1.0 //depot/release/jam/2.1/src/*.h
```

タグの結果をプレビューする

p4 tag の結果は、p4 tag -n を使ってプレビューすることができます。このコマンドでは、実際に操作を行うことなく、tag コマンドによってタグ付け、タグの取り外し、再度のタグ付けを行うリビジョンをリスト表示します。

ラベルによってタグ付けされたファイルをリストする

labelname によりタグ付けされたリビジョンをリストするには、次のようにラベル名を指定して p4 files を使用します。

```
p4 files @labelname
```

labelname でタグ付けされたすべてのリビジョンが、ファイルタイプ、チェンジの操作、チェンジリスト番号と共にリストされます。(このコマンドは、p4 files //...*@labelname* と同等です。)

ファイルに適用済みのラベルをリストする

任意のファイルに適用されたすべてのラベルをリストするには、次のコマンドを使用します。

```
p4 labels filepattern
```

ラベルを使ってファイル・リビジョンを指定する

ファイルを参照するためにリビジョン (#1, #head)、チェンジリスト番号 (@7381)、または日付 (@2003/07/01) を指定できる箇所であれば、ラベル名も指定することができます。

p4 sync *@labelname* コマンドの発行時にファイル引数を省略すると、そのラベルによってタグ付けされたクライアント・ワークスペース・ビュー内のすべてのファイルは、ラベルで指定されたリビジョンに同期されます。ラベルでタグ付けされているリビジョンを含まないワークスペース内のすべてのファイルは、ワークスペースから削除されます。作業状態のファイルと PERFORCE の管理下にないファイルは、その影響を受けません。このコマンドは、p4 sync //...*@labelname* と同等です。

p4 sync コマンドの発行時にファイル引数を指定すると (p4 sync *files@labelname*)、ワークスペースにある、ラベルでタグ付けされたファイルがタグ付けされたリビジョンに同期されます。

例: *ラベルによってタグ付けされたファイルをクライアント・ワークスペースに取得する*

アールのラベル jam-2.1.0 *によってタグ付けされたファイルを彼のクライアント・ワークスペースに取得するために、ブルーノは次のコマンドを発行します。*

```
p4 sync @jam-2.1.0
```

結果として、次のように出力されます。

```
//depot/dev/main/jam/Build.com#5 - updating c:¥bruno_ws¥dev¥main¥jam¥Build.com
//depot/dev/main/jam/command.c#5 - updating c:¥bruno_ws¥dev¥main¥jam¥command.c
//depot/dev/main/jam/command.h#3 - added as c:¥bruno_ws¥dev¥main¥jam¥command.h
//depot/dev/main/jam/compile.c#12 - updating c:¥bruno_ws¥dev¥main¥jam¥compile.c
//depot/dev/main/jam/compile.h#2 - updating c:¥bruno_ws¥dev¥main¥jam¥compile.h
〈など 〉
```

ラベルを削除する

ラベルを削除するには、次のコマンドを使用します。

```
p4 label -d labelname
```

ラベルを削除しても、タグ付きのファイル・リビジョンには影響しません（ただし、当然ながら当該リビジョンのタグはなくなります)。

今後のためにラベルを作成する

どのファイル・リビジョンにもタグを付けずにラベルを作成するには、p4 label *labelname* コマンドを発行します。このコマンドは、ラベルの説明の記述や指定に使われるフォームを表示します。ラベルを作成したら、p4 tag または p4 labelsync を使ってそのラベルをファイル・リビジョンに適用できます。

ラベル名は、クライアント・ワークスペース、ブランチ、またはディポの名前と異なっていなければなりません。

例えば、jam-2.1.0 を作成するには、次のコマンドを発行します :

```
p4 label jam-2.1.0
```

次のフォームが表示されます。

```
Label:    jam-2.1.0
Update:    2005/03/07 13:07:39
Access:    2005/03/07 13:13:35
Owner:    earl
Description:
     Created by earl.
Options:     unlocked
View:
     //depot/...
```

ラベルの説明を入力し、フォームを保存します（[View:] フィールドを変更する必要はありません)。

ラベルを作成した後は、p4 tag および p4 labelsync コマンドを使ってそのラベルをファイル・リビジョンに適用できます。

タグ付け可能なファイルを制限する

p4 label フォーム内の [View:] フィールドは、ラベルでタグ付けができるファイルを制限します。デフォルト・ラベル・ビューにはディポ全体(//depot/...)が含まれます。ディポ内のすべてのファイルに対して不用意にタグ付けをしないようにするには、ラベルの [View:] フィールドでディポ・シンタックスを使用して、タグ付け可能なファイルとディレクトリを設定します。

例 : *ラベル・ビューを使用してタグ付け可能なファイルを制御する*

アールは、リリース 2.1 ブランチ内のソースコードのリビジョンにタグ付けしたいと考えています。彼はそれらのリビジョンが正常にコンパイルできることを知っています。彼は p4 label jam-2.1.0 *と入力し、ラベルの適用範囲を制限するため、以下のようにラベルの [View:] フィールドを指定します。*

```
Label:    jam-2.1.0
Update:    2005/03/07 13:07:39
Access:    2005/03/07 13:13:35
Owner:    earl
Description:
     Created by earl.
Options:     unlocked
View:
     //depot/release/jam/2.1/src/...
```

このラベルは、release 2.1 ソースコード・ディレクトリにあるファイルにのみタグ付けすることができます。

静的ラベルを使ってワークスペースの設定を記録する

p4 labelsync コマンドを発行することにより、静的ラベルを使用してクライアント・ワークスペースの状態（現在同期されているファイル・リビジョン）を記録することができます。指定するラベルは、クライアント・ワークスペースと同じビューを持っていなければなりません。

例えば、既存の ws_config というラベルを使って現在のクライアント・ワークスペースの設定を記録するには、次のコマンドを使用します。

```
p4 labelsync -l ws_config
```

現在のワークスペースに同期されていてクライアント・ビューとラベル・ビュー（存在すれば）の両方から見ることのできるすべてのファイル・リビジョンに、ws_config というラベルがタグ付けされます。それまで ws_config がタグ付けされていて、その後ワークスペースから削除された（p4 sync #none）ファイルからは、タグが外されます。

ws_config ラベルでタグ付けされたファイルを同期させる（その結果、ワークスペースの設定を再作成する）には、次のコマンドを発行します。

```
p4 sync @ws_config
```

自動ラベルをチェンジリストや他のリビジョンの別名として使用する

自動ラベルを使用すれば、p4 labelsync コマンドを発行せずにファイルの特定のリビジョンを指定することができます。

自動ラベルを作成するには、p4 label フォームの [Revision:] フィールドにリビジョン指定子を入力します。ワークスペースを自動ラベルに同期させると、[Revision:] フィールドの内容が [View:] フィールドのすべてのファイルに適用されます。

例 : *自動ラベルをチェンジリスト番号の別名として使用する*

アールは毎晩ビルド処理を実行しており、チェンジリスト 1234 として製品の構築に成功しました。毎晩のビルドごとに固有のチェンジリストを覚えておく必要がないよう、アールは p4 label nightly20061201 *と入力してラベルの* [Revision:] *フィールドを使用し、自動的にチェンジリスト 1234 のすべてのファイルに* nightly20061201 *ラベルがタグ付けされるようにします。*

```
Label:    nightly20061201
Owner:    earl
Description:
     Nightly build process.
Options:     unlocked
View:
     //depot/...
Revision:
     @1234
```

この方法の利点は、スクリプト作成に非常に適しており、ラベル・テーブルの領域をほとんど占有せず、毎晩のビルドを簡単に参照する方法を提供していることです。それにより、どのチェンジリスト番号がどの晩のビルド処理に関連付けられているかを覚えておく必要はありません。

例 : *ある 1 つのチェンジリストでサブミットされた一連のファイルを限定的に参照する*

あるバグがチェンジリスト 1238 により修正されており、その修正に関連するファイルのみを参照するパッチ・ラベルが必要です。アールは p4 label patch20061201 *と入力してラベルの* [Revision:] *フィールドを使用し、自動的にチェンジリスト 1238 でサブミットされたファイルにのみ* patch20061201 *ラベルがタグ付けされるようにします。*

```
Label:    nightly20061201
Owner:    earl
Description:
     Patch to 2006/12/01 nightly build.
Options:     unlocked
View:
     //depot/...
Revision:
     @1238,1238
```

この自動ラベルは、チェンジリスト 1238 でサブミットされたファイルのみを参照しています。

例 : *複数のチェンジリストにある各ファイルについて最初のリビジョンを参照する*

チェンジリスト指定子以外にリビジョン指定子を使用することができます。この例では、アールはブランチ内の各ファイルについて最初のリビジョン (#1) を参照しています。ブランチの実装方法によって、これらのファイルは長い期間を経て複数のチェンジリストから作成された可能性があります。

```
Label:    first2.2
Owner:    earl
Description:
     The first revision in the 2.2 branch
Options:     unlocked
View:
     //depot/release/jam/2.2/src/...
Revision:
     "#1"
```

PERFORCE のフォームでは、# はコメントを表すために使われるため、アールは # を二重引用符で囲み、この文字がリビジョン指示子として解析されるようにしています。

ファイルへの誤ったタグ付けおよび取り外しを防ぐ

クライアント・ワークスペースおよびラベル・ビュー（設定している場合）にあるファイルにタグ付けし、それ以外のファイルからはタグを取り外すには、引数を指定せずに `p4 labelsync` コマンドを発行します。ファイルへの誤ったタグ付けおよび取り外しを防ぐには、`p4 label` *`labelname`* コマンドを発行し、ラベル・フォームの `[Options:]` フィールドの値を `locked` に設定することによって、ラベルをロックします。他のユーザがラベルのロックを解除できないようにするには、`[Owner:]` フィールドを設定します。PERFORCE の権限について詳しくは、*『PERFORCE システム管理者ガイド』* を参照してください。

第 7 章

欠陥追跡

ジョブ とは、PERFORCE サーバにより管理される番号付きの（または名前が付けられた）作業要求です。PERFORCE ジョブにより、バグや改善要求のステータスを追跡し、修正および改善を実装するチェンジリストにそれらを関連付けることができます。ジョブは、フィールドの内容、ジョブの入力日付または最終更新日付、その他多くの検索基準に基づいて検索することができます。

PERFORCE 管理者は、サイトでの要求に従ってジョブ仕様をカスタマイズできます。ジョブ仕様の変更について詳しくは、*『PERFORCE システム管理者ガイド』* をご覧ください。

PERFORCE を社内で稼動している欠陥追跡システムと統合したい場合、またはサード・パーティの欠陥追跡システムとの統合を開発したい場合は、PERFORCE Web サイトの P4DTI 製品情報ページをご覧ください。

ジョブの管理

PERFORCE のデフォルトのジョブ命名スキームを使用してジョブを作成するには、`p4 job` コマンドを発行します。新しいジョブに名前を割り当てる（または既存のジョブを編集する）には、`p4 job` *`jobname`* コマンドを発行します。

例: *ジョブの作成*

ゲイルは Jam に関する問題を発見したので、`p4 job` コマンドを発行してジョブを作成し、その説明を次のように記述します。

```
Job:     job000006
Status:     open
User:     gale
Date:     2005/11/14 17:12:21
Description:
     MAXLINE on NT can't account for NT 4.0 expanded cmd buffer size.
```

以下の表に、デフォルトのジョブ仕様にあるフィールドを記述します。

フィールド名	説明	デフォルト
`Job`	ジョブの名前 (空白は不可)。デフォルトでは、PERFORCE は番号を付ける方式でジョブ名を割り当てます (`job`*`nnnnnn`*)。	最新のジョブ番号 + 1
`Status`	• `open`: 修正が済んでいないジョブ。 • `closed`: 完了したジョブ。 • `suspended`: 現在作業が保留されているジョブ。	`open`
`User`	ジョブが割り当てられているユーザ。通常、この問題の解決にあたる担当者のユーザ名。	ジョブ作成者の PERFORCE でのユーザ名

フィールド名	説明	デフォルト
Date	ジョブを最後に変更した日付。	ジョブの保存時に PERFORCE サーバにより更新される。
Description	要求されている作業の説明。例えば、バグの説明やシステム改善要求。	なし。説明の記述は必須。

既存のジョブを編集するには、p4 job コマンドの発行時に p4 job *jobname* の形式でジョブ名を指定します。加えた変更をジョブ・フォームに入力し、フォームを保存して終了します。

ジョブを削除するには、p4 job -d *jobname* コマンドを発行します。

ジョブの検索

PERFORCE ジョブを検索するには、p4 jobs -e *jobview* コマンドを発行します。*jobview* には、以降のセクションで記述する検索式を指定します。詳しくは、p4 help jobview コマンドを発行してください。

ジョブ・テキストを検索する

'word1 word2 ... wordN' という式を使用して、いずれかのフィールド（日付フィールドを除く）に *word1* から *wordN* のすべてを含むジョブを検索することができます。UNIX では単一引用符を、Windows では二重引用符を使用してください。

ジョブを検索するときは、以下の制限事項に注意してください。

- 複数の単語を空白で区切って指定すると、PERFORCE は指定された*すべての*単語を含むジョブを検索します。単語のうち*どれかを*含むジョブを検索するには、単語をパイプ（|）で区切ってください。
- 式の中にあるフィールド名と比較テキストでは、大文字と小文字は区別されません。
- 英数字テキストおよび句読文字に限り、式に表すことができます。PERFORCE により論理演算子として使用される以下の文字と照合するときは、その前にバックスラッシュを付けてください。=^&|()<>
- 語句（フレーズ）を検索することはできません。単語のみ検索できます。

例: *指定の単語を含むジョブの検索*

*ブルーノは、*filter、file、mailbox *という語を含むすべてのジョブを検索したいと考え、次のように入力します。*

```
p4 jobs -e 'filter file mailbox'
```

例: *いずれかのフィールドに一連の単語のうちどれかを含むジョブの検索*

*ブルーノは、*filter、file、mailbox *という単語のどれかを含むジョブを検索したいと考え、次のように入力します。*

```
p4 jobs -e 'filter|file|mailbox'
```

ワイルドカード「*」を使用して、1 つ以上の文字と照合することができます。例えば fieldname=string* という式には、string、strings、stringbuffer などが適合します。

ワイルドカードに使用される文字を含む単語を検索するには、コマンドにおいてワイルドカード文字の前にバックスラッシュを付けます。例えば、*string という語（おそらく char *string を参照）を検索するには、次のコマンドを発行します。

```
p4 jobs -e '\*string'
```

特定のフィールドの検索

特定フィールドの値に基づいて検索するには、*field=value* を指定します。

例: *特定フィールド内に指定の単語を含むジョブの検索*

ブルーノは、フィルタリングに関連する作業状態のジョブをすべて検索したいと考え、次のように入力します。

```
p4 jobs -e 'Status=open User=bruno filter.c'
```

このコマンドにより、[Status:] *(状態) フィールドが* open、[User:] *(ユーザ) フィールドが* bruno *で、日付フィールド以外のどこかに単語* filter.c *を含むすべてのジョブが検索されます。*

指定された表現を含まないフィールドを検索するには、その前に否定演算子である ^ を付けます。否定演算子 ^ は、アンド式（スペースまたは &）の直後にしか使用できません。例えば、p4 jobs -e '^user=bruno' は使用できません。このような制限を回避するには、ワイルドカード「*」を使用して、^ を付けた語の前に検索語を追加します。例えば、p4 jobs -e 'job=* ^user=bruno' とするとブルーノが所有していないすべてのジョブが返されます。

例: *あるフィールドに特定の値が含まれているジョブを除外する*

ブルーノは、フィルタリングに関連し、自分が所有者でない作業状態のジョブをすべて検索したいと考え、次のように入力します。

```
p4 jobs -e 'status=open ^user=bruno filter'
```

*このコマンドにより、ブルーノが所有していない、*filter *という単語を含む作業状態のジョブがすべて表示されます。*

比較演算子の使用

以下に示す比較演算子が使用できます。

=	等価
>	より大きい
<	より小さい
>=	以上
<=	以下

これらの演算子の動作は、式の中にあるフィールドのタイプにより異なります。以下の表に、フィールド・タイプおよびそのタイプがどのように検索されるかを示します。

フィールド・タイプ	説明	注意
word	1 つの単語	等価演算子 (=) には、指定フィールドの値が、指定した単語に完全に一致するジョブが適合します。 関係演算子は、ASCII の順番に沿って比較を実行します。
text	フィールド名の下の行に入力された、1 つのテキストのまとまり	等価演算子 (=) には、指定フィールドに指定された値が含まれているジョブが適合します。 このフィールド・タイプにおける関係演算子の使用は、限定的なものです。指定したフィールド内の単語のうち、*1* つでも指定した値と合致すれば、そのジョブが適合することになるからです。例えば、gui bug という語句のみを含む ShortDescription: というテキスト・フィールド を持つジョブがあり、式が 'ShortDesc<filter' である場合、bug<filter なので、このジョブはその式に適合します。
line	フィールド名と同一の行に入力された、1 行のテキスト	text と同じ
select	1 組の値セットのうち 1 つ。例えば、ジョブの状態は open/suspended/closed のどれかになります。	等価演算子 (=) には、フィールドの値が特定の単語であるジョブが適合します。 関係演算子は、ASCII の順番に沿って比較を実行します。
date	日付および任意指定の時刻。例えば、2005/07/15:13:21:40	日付は時系列で適合します。特定の時刻を指定しない場合、演算子 =、<=、および >= によって、その日の時刻すべてが適合します。

フィールド・タイプが分からない場合、p4 jobspec -o コマンドを発行すればジョブ仕様が表示されます。[Fields:] フィールドに、ジョブ・フィールドの名前とデータ・タイプが一覧表示されます。

日付フィールドの検索

日付フィールドを検索するには、*yyyy/mm/dd* または *yyyy/mm/dd:hh:mm:ss* の書式を使用して日付を指定します。特定の時刻を指定しない場合、等価演算子 (=) にはその日のすべての時刻が適合します。

例 : *式の中での日付の使用*

ブルーノは 2005 年 7 月 13 日に変更したすべてのジョブを参照したいと考え、次のように入力します。

```
p4 jobs -e 'ModifiedDate=2005/07/13'
```

ジョブの解決

ジョブを修正するには、そのジョブをチェンジリストにリンクして、チェンジリストをサブミットします。チェンジリストがサブミットされると、PERFORCE は自動的にジョブの [Status:] フィールドの値を closed に変更します。

ジョブをチェンジリストにリンクする方法には、以下の 3 つがあります。

- p4 user フォームの [JobView:] フィールドに、ジョブに適合する式を設定します。
- p4 fix コマンドを使用します。
- p4 submit フォームを編集します。

ジョブを編集することにより直接ジョブ・ステータスを変更できますが、手動でジョブをクローズすると、そのジョブを修正したチェンジリストとの関連付けが失われます。[Status:] フィールドを削除して自分のサイトのジョブ仕様を変更している場合、ジョブをチェンジリストにリンクすることは可能ですが、チェンジリストがサブミットされていると、ステータスは変更できません。(多くの場合、これは期待される処理方法ではありません。) 詳しくは、*『PERFORCE システム管理者ガイド』* のジョブ仕様の編集に関する章をご覧ください。

ジョブをチェンジリストから削除するには、p4 fix -d コマンドを発行します。

自動的にリンクする

PERFORCE ユーザ仕様を変更して、自動的に作業状態のジョブを作成した任意のチェンジリスト上に組み込むことができます。自動組み込みを設定するには、p4 user コマンドを発行し、[JobView:] フィールドの値に組み込みたいジョブを特定する有効な式を設定します。

例 : *自動的にジョブをチェンジリストにリンクする*

*ブルーノは、自分が作成したすべてのチェンジリスト内で自分が所有している作業状態のジョブをすべて参照したいと考え、*p4 user *と入力し、*[JobView:] *フィールドに次のように追加します。*

```
User:     bruno
Update:   2005/06/02 13:11:57
Access:   2005/06/03 20:11:07
JobView:  user=bruno&status=open
```

これでブルーノが所有している作業状態のジョブがすべて、自動的に彼のデフォルト・チェンジリストに組み込まれます。チェンジリストをサブミットするとき、ブルーノは彼がサブミットするチェンジリストで修正されていないジョブを確実に削除しなければなりません。

手動でリンクする

ジョブを手動でチェンジリストにリンクするには、`p4 fix -c changenum jobname` コマンドを発行します。そのチェンジリストが既にサブミットされていれば、ジョブの `[Status:]` フィールドの値は `closed` に変化します。それ以外の場合は、ジョブのステータスは変わりません。

例: *手動でジョブをチェンジリストにリンクする*

`p4 fix` を使用して、チェンジリストを他のユーザが所有するジョブにリンクすることができます。

サラは `options-bug` というジョブをブルーノに割り当てたところですが、そのバグはすでに、ブルーノが以前にサブミットしたチェンジリスト 18 で修正されていました。ブルーノは次のように入力して、ジョブをそのチェンジリストにリンクします。

```
p4 fix -c 18 options-bug
```

チェンジリスト 18 は既にサブミットされているので、ジョブの状態は `closed`. に変わります。

ジョブをチェンジリストにリンクする

チェンジリストをサブミットまたは編集する際にジョブをチェンジリストにリンクするには、チェンジリスト仕様の `[Jobs:]` フィールドにジョブ名を入力します。チェンジリストをサブミットすると、ジョブは（デフォルトでは）クローズされます。

作業中チェンジリストへのジョブのリンクを解除するには、チェンジリストを編集して、`[Jobs:]` フィールドからジョブの名前を削除します。サブミット済チェンジリストへのジョブのリンクを解除するには、`fix -d -c changenum jobname` コマンドを発行します。

第 8 章

スクリプトとレポートに使用するコマンド

本章では p4 コマンドのスクリプト内での使用法、およびレポート目的での使用法について詳しく説明します。特定のコマンドに関する詳細な説明は、*『PERFORCE コマンド・リファレンス』*をご覧になるか、p4 help コマンドを発行してください。

スクリプトとレポートの共通オプション

以下のコマンドライン・オプションにより、コマンドライン上およびスクリプト内で設定を指定することができます。詳細な情報は、*『PERFORCE コマンド・リファレンス』*のグローバル・オプションに関する説明を参照してください。

オプション	説明
-c *client_workspace*	クライアント・ワークスペース名を指定します。
-G	すべての出力（およびフォーム・コマンドに対して -i で指定する一括入力）を、整列された Python ディレクトリ・オブジェクトとしてフォーマットします。
-p *server*:*port*	PERFORCE サーバのホストおよびポート番号を指定します。
-P *password*	設定されている場合、ユーザ・パスワードを指定します。（コマンドを発行するたびにパスワードを指定する代わりに）スクリプトでコマンドの実行前にログインするようにしたい場合、p4 login コマンドを使用します。 例 :echo 'mypassword' | p4 login
-s	PERFORCE コマンドにより生成された出力の各行の最初に、説明的なフィールド（text:、info:、error:、exit: など）を追加します。
-u *user*	PERFORCE ユーザ名を指定します。
-x *argfile*	指定のファイルから 1 行ごとに 1 つずつ、引数を読み込みます。*argfile* が単一のハイフン（-）である場合、標準入力から読み込まれます。

PERFORCE フォームによるスクリプト記述

スクリプトが p4 client や p4 submit などのようにユーザにフォームへの入力を要求するコマンドを発行する場合、-o オプションを使用して標準出力にフォームを書き込み、-i オプションを使用して標準入力から編集後のフォームを読み込みます。

例えば、UNIX のスクリプトを使用してジョブを作成するには、以下のようにします。

1. p4 job -o > temp1 コマンドを発行し、空白のジョブ仕様をテキスト・ファイルに書き込みます。
2. ジョブに必要な変更を加えます。

   例:sed 's/<enter description here>/Crash when exiting./' temp1 > temp2
3. p4 job -i < temp2 コマンドを発行してジョブを保存します。

一時ファイルを作成せずに上記の処理を実行するには、次のコマンドを発行します。

```
p4 job -o | sed 's/<enter description here>/Crash when exiting./' | p4 job -i
```

フォームを表示するコマンドは以下のとおりです。

- p4 branch
- p4 change
- p4 client
- p4 job
- p4 label
- p4 submit (チェンジリストの作成には p4 change -o を使用。チェンジリストのサブミット時にデフォルト・チェンジリストにコメントを入れるには、p4 submit -d "A changelist description" を使用。)
- p4 user

ファイルのレポート

次のセクションでは、ファイルの状態と場所に関する情報を提供するコマンドを説明します。次の表に、基本的で非常に有用なレポート・コマンドを示します。

表示させたい内容	使用するコマンド
ファイルの状態。ファイルタイプ、最新リビジョン番号、その他の情報など。	p4 files
ファイル・リビジョンを、最新のものから最も古いものまで表示	p4 filelog
現在作業状態のファイル	p4 opened
p4 sync の結果をプレビュー	p4 sync -n
現在同期されているファイル	p4 have
指定されたファイルの内容	p4 print
ファイルのディポでの場所と対応するワークスペースの場所とのマッピング	p4 where
ファイルの一覧およびファイルに関する詳細情報	p4 fstat

ファイルの状態を表示する

ファイルの 1 つのリビジョンの情報を表示するには、p4 files コマンドを発行します。このコマンドはディポ内でのファイルの場所、指定リビジョンでそれらのファイルに行われた動作 (add、edit、delete など)、指定ファイル・リビジョンがサブミットされたチェンジリスト、およびファイルのタイプを表示します。p4 files コマンドの出力例を示します。

```
//depot/README#5 - edit change 6 (text)
```

p4 files コマンドには1つ以上の *filespec* 引数が必要です。*filespec* 引数の指定にローカル・シンタックス、クライアント・シンタックス、ディポ・シンタックスのうちどれを使用するかに関係なく、p4 file コマンドはディポ・シンタックスを使用して結果を表示します。リビジョン番号を指定しない場合、最新リビジョンの情報を表示します。p4 files の出力には削除されたリビジョンも含まれます。

次の表に、p4 files コマンドの一般的な使用方法をいくつか示します。

表示させたい状態	使用するコマンド
クライアント・ワークスペース・ビューに関係なく、ディポ内のすべてのファイル。	p4 files //depot/...
非常に多くのファイルを含むディポの場合、必要がない限りディポ全体 (//depot/...) を参照するコマンドを発行しないようにしてパフォーマンスを最良にすることができます。パフォーマンスを最良にするため、目的のディレクトリおよびファイルのみを指定するようにしてください。	
指定されたクライアント・ワークスペースに現在同期させているファイル	p4 files @*clientname*
自分のクライアント・ワークスペース・ビューにマッピングされているファイル	p4 files //*clientname*/...
現在作業中のディレクトリにある特定のファイル	p4 files *filespec*
特定のファイル・リビジョン	p4 files *filespec*#*rev*
ファイルがそのチェンジリストでサブミットされたかどうかに関わらず、チェンジリストがサブミットされた時点の特定のファイル	p4 files *filespec*@*changenum*
特定のラベルでタグが付けられているファイル	p4 files *filespec*@*labelname*

ファイルのリビジョン履歴の表示

ファイルのリビジョン履歴を表示するには、p4 filelog *filespec* コマンドを発行します。以下の例に、p4 filelog によりどのようにリビジョン履歴が表示されるかを示します。

```
p4 filelog //depot/dev/main/jam/jam.c
//depot/dev/main/jam/jam.c
... #35 change 627 edit on 2001/11/13 by earl@earl-dev-yew (text)
'Handle platform variants better'
... #34 change 598 edit on 2001/10/24 by raj@raj-althea (text) 'Reverse
previous attempt at fix'
... ... branch into //depot/release/jam/2.2/src/jam.c#1
... #33 change 581 edit on 2001/10/03 by gale@gale-jam-oak (text)
'Version strings & release notes'
```

各チェンジリストの説明をすべて表示するには、-l オプションを指定します。

作業状態のファイルを一覧表示する

現在クライアント・ワークスペース内で作業状態となっているファイルを一覧表示するには、p4 opened *filespec* コマンドを発行します。次に示す行は、p4 opened コマンドによる出力表示の例です。

```
//depot/dev/main/jam/fileos2.c- edit default change (text)
```

次の表に、p4 opened コマンドの一般的な使用方法をいくつか示します。

一覧表示する内容	使用するコマンド
現在のワークスペースで作業状態になっているファイル	p4 opened
すべてのクライアント・ワークスペースにある作業状態のファイル	p4 opened -a
番号付きの作業中チェンジリストに含まれているファイル	p4 opened -c *changelist*
デフォルトチェンジリストに含まれているファイル	p4 opened -c *default*
特定のファイルが自分のワークスペースで作業状態になっているかどうか	p4 opened *filespec*
特定のファイルが作業状態になっているかどうか	p4 opened -a *filespec*

ファイルの場所を表示する

ファイルの場所に関する情報を表示するには、p4 where、p4 have、および p4 sync -n の各コマンドを使用します。

- ディポ・シンタックス、クライアント・シンタックス、ローカル・シンタックスでファイルの場所を表示するには、p4 where コマンドを発行します。
- 自分のクライアント・ワークスペースに最後に同期させたファイルの場所およびリビジョンをリスト表示するには、p4 have コマンドを発行します。
- ファイルが自分のワークスペース内のどこに同期されたかを調べるには、p4 sync -n コマンドを発行して同期結果をプレビューします。

上記のコマンドでは、*filespec* 引数を付けても付けなくても使用できます。

次の表に、ファイルの場所を知るための有用なコマンドをいくつか示します。

表示する内容	使用するコマンド
自分のワークスペースに同期させているファイルのリビジョン番号	p4 have *filespec*
ディポ内にある特定のファイルが自分のワークスペースのどこにマッピングされているか	p4 where //depot/*filespec*

ファイルの内容を表示する

ディポ内のファイルの内容を表示するには、p4 print *filespec* コマンドを発行します。このコマンドは、ファイルを説明する 1 行のヘッダとともに、ファイルの内容を標準出力または指定出力ファイルに出力します。-q オプションを指定すれば、ヘッダは出力されません。デフォルトでは最新リビジョンが表示されますが、ファイル・リビジョンを指定することもできます。

表示させたいファイル内容	使用するコマンド
最新リビジョン	p4 print *filespec*
ヘッダを含まないファイル内容	p4 print -q *filespec*
特定のチェンジリスト番号のファイル	p4 print *filespec*@*changenum*

注釈（ファイル内容への変更の詳細情報）を表示する

テキストファイルのどの行をどのファイル・リビジョンまたはチェンジリストで変更したかを調べるには、p4 annotate コマンドを発行します。

デフォルトでは、p4 annotate はファイルの各行を表示し、変更したリビジョンがわかるように、各行の先頭にリビジョン番号が付加されます。リビジョン番号ではなくチェンジリスト番号を表示するには、-c オプションを指定します。

例: p4 annotate *を使って、ファイルへの変更を表示*

*ディポにファイル (*file.txt#1*) が追加され、ファイルには以下の行が含まれています。*

```
This is a text file.
The second line has not been changed.
The third line has not been changed.
```

*3 行目が削除され、2 行目が編集されて、*file.txt#2 *は次のようになります。*

```
This is a text file.
The second line is new.
```

p4 annotate *および* p4 annotate -c *の出力は次のようになります。*

```
$ p4 annotate file.txt
//depot/files/file.txt#3 - edit change 153 (text)
1: This is a text file.
2: The second line is new.
$ p4 annotate -c file.txt
//depot/files/file.txt#3 - edit change 153 (text)
151: This is a text file.
152: The second line is new.
```

file.txt *の最初の行はリビジョン 1 から存在し、このリビジョンはチェンジリスト 151 でサブミットされています。2 行目はリビジョン 2 から存在し、そのリビジョンはチェンジリスト 152 でサブミットされています。*

このファイルのすべての行を（削除された行を含め）表示するには、次のように p4 annotate -a *を実行します。*

```
$ p4 annotate -a file.txt
//depot/files/file.txt#3 - edit change 12345 (text)
1-3: This is a text file.
1-1: The second line has not been changed.
1-1: The third line has not been changed.
2-3: The second line is new.
```

*出力の最初の行は、ファイルの 1 行目がリビジョン 1 から 3 まで存在していることを示しています。次の 2 行は、*file.txt *のこれらの行がリビジョン 1 にのみ存在したことを示しています。最後の行は、リビジョン 2 で追加された行がリビジョン 3 にも存在していることを示しています。*

-a *オプションと* -c *オプションを組み合わせると、これまでこのファイルに存在したすべての行と、各行が存在したチェンジリスト番号（リビジョン番号ではない）を表示することができます。*

ファイルへの変更を監視する

ファイルへの変更を発生の都度追跡するために、PERFORCE のチェンジ・レビュー・デーモンを使用できます。これにより、PERFORCE ユーザが目的のファイルやディレクトリを指定したり、特定のファイルを変更するチェンジリストがサブミットされたときにメールを受信することができます。レビュー・デーモンの管理に関する詳細情報は、*『PERFORCE システム管理者ガイド』*、および *『PERFORCE コマンド・リファレンス』* の p4 review コマンドの説明を参照してください。

次の表に、ファイル、チェンジリスト、およびユーザの状態に関する情報を表示するコマンドを示します。これらのコマンドはレビュー・デーモンでよく使用されます。

一覧表示したい内容	使用するコマンド
特定のファイルを閲覧するユーザ	p4 reviews *filespec*
特定のチェンジリスト内のファイルを閲覧するユーザ	p4 reviews -c *changenum*
特定のユーザの電子メール・アドレス	p4 users *username*

チェンジリストのレポート

p4 changes コマンドは特定の検索基準に合うチェンジリストを一覧表示し、p4 describe コマンドは特定のチェンジリストに関連付けられているファイルやジョブを一覧表示します。以下、これらのコマンドについて解説します。

チェンジリストを一覧表示する

チェンジリストを一覧表示するには、p4 changes コマンドを発行します。デフォルトでは、p4 changes はシステムが認識しているすべてのチェンジリストについてそれぞれ1行ずつ表示します。以下の表に、リストのフィルタリングに使用できるコマンドライン・オプションを示します。

一覧表示させたいチェンジリスト	使用するコマンド
チェンジリストの記述の最初の 31 文字を表示	p4 changes
チェンジリストの完全な記述を表示	p4 changes -l
最新の *n* 個のチェンジリスト	p4 changes -m n
特定の状態のチェンジリスト	p4 changes -s pending または p4 changes -s submitted
特定ユーザに関連するチェンジリスト	p4 changes -u *user*
特定のワークスペースに関連するチェンジリスト	p4 changes -c *workspace*
特定ファイルに関連するチェンジリスト	p4 changes *filespec*
特定ファイルに関連するチェンジリストを示すが、特定ファイルの反映元となったファイルに関連するチェンジリストを含める	p4 changes -i *filespec*
特定ファイルに関連するチェンジリストを示し、これらのファイルのリビジョン *m* とリビジョン *n* の間のチェンジリストのみを含める	p4 changes *filespec*#*m*,#*n*
ラベル *lab1* とラベル *lab2* の間の各リビジョンで、特定ファイルに関連するチェンジリスト	p4 changes *filespec*@*lab1*,@*lab2*
2 つの日付の間にサブミットされたチェンジリスト	p4 changes @*date1*,@*date2*
特定の日以降にサブミットされたチェンジリスト	p4 changes @*date1*,@*now*

チェンジリストに関連するファイルとジョブを一覧表示する

特定のチェンジリストに関連するファイルとジョブの一覧を、変更の差分とともに表示するには、p4 describe コマンドを発行します。差分を省略する（短い形式の出力）には、-s オプションを指定します。次の表に、チェンジリストのレポートに使われる有用なコマンドをいくつか示します。

一覧表示させたい内容	使用するコマンド
作業中チェンジリストに含まれるファイル	p4 opened -c *changenum*
特定のチェンジリストでサブミットされたファイル、および修正されたジョブ。差分を含む。	p4 describe *changenum*
特定のチェンジリストでサブミットされたファイル、および修正されたジョブ。差分は表示しない。	p4 describe -s *changenum*
特定のチェンジリストに関連するファイルおよびジョブ。コンテキスト差分オプションを基盤となる比較プログラムに渡す。	p4 describe -dc *changenum*
特定のチェンジリストにおける特定ファイルの状態を、そのチェンジリストがこれらのファイルに関連するかどうかに関わらず表示	p4 files *filespec*@*changenum*

ジョブについてレポートするその他のコマンドに関しては、91 ページの「ジョブのレポート」をご覧ください。

ラベルのレポート

ラベルに関する情報を表示するには、p4 labels コマンドを発行します。次の表に、ラベルのレポートに使われる有用なコマンドをいくつか示します。

一覧表示させたい内容	使用するコマンド
すべてのラベルとその作成日付および所有者	p4 labels
指定したファイル・リビジョン（またはリビジョン範囲）を含むすべてのラベル	p4 labels file#*revrange*
特定のラベルでタグ付けされたファイル	p4 files @*labelname*
ラベルを同期させた結果のプレビュー	p4 sync -n @*labelname*

ブランチと反映のレポート

以下の表に、ブランチおよび反映操作に関するレポートに一般的に使用されるコマンドを示します。

一覧表示させたい内容	使用するコマンド
すべてのブランチ仕様	p4 branches
特定のブランチ内のファイル	p4 files *filespec*
特定ファイルのリビジョン	p4 filelog *filespec*
特定ファイルのリビジョン。反映元となったファイルのリビジョンを再帰的に含む。	p4 filelog -i *filespec*
衝突解決の結果のプレビュー表示	p4 resolve [*args*] -n [*filespec*]
解決されたがまだサブミットされていないファイル	p4 resolved [*filespec*]
反映およびサブミットされているファイルのうち、*filespec* 引数に適合するファイル	p4 integrated *filespec*
反映操作の結果のプレビュー表示	p4 integrate [*args*] -n [*filespec*]

ジョブのレポート

ジョブを一覧表示する

ジョブを一覧表示するには、p4 jobs コマンドを発行します。以下の表に、ジョブのレポートに一般的に使用されるコマンドを示します。

一覧表示させたい内容	使用するコマンド
すべてのジョブ	p4 jobs
すべてのジョブに、ジョブの完全な説明を含む	p4 jobs -l
検索基準に合致するジョブ（詳しくは 78 ページの「ジョブの検索」を参照）	p4 jobs -e *jobview*
特定のファイルを含むチェンジリストで修正されたジョブ	p4 jobs *filespec*
特定のファイルを含むチェンジリストで修正されたジョブ。その特定のファイルの反映元となったファイルに関連するチェンジリストを含める	p4 jobs -i *filespec*

チェンジリストにより解決したジョブを一覧表示する

p4 change、p4 submit、および p4 fix でチェンジリストにリンクされたジョブは、「*修正されたジョブ*」(それらの状態が closed であるかどうかにかかわらず) と呼ばれます。チェンジリストにより修正されたジョブを一覧表示するには、p4 fixes コマンドを発行します。

以下の表に、修正のレポートに使われる有用なコマンドを示します。

一覧表示させたい内容	使用するコマンド
ジョブにリンクされているすべてのチェンジリスト	p4 fixes
特定のジョブにリンクされているすべてのチェンジリスト	p4 fixes -j *jobname*
特定のチェンジリストにリンクされているすべてのジョブ	p4 fixes -c *changenum*
特定のファイルに関連するすべての修正	p4 fixes *filespec*
特定のファイルに関連するすべての修正。その特定のファイルの反映元となったファイルに関連するチェンジリストを含める。	p4 fixes -i *filespec*

システム構成のレポート

このセクションで説明するコマンドでは、PERFORCE ユーザ、クライアント・ワークスペース、およびディポが表示されます。

ユーザの表示

p4 users コマンドはユーザ名、電子メール・アドレス、ユーザの「本」名、およびユーザが最後に PERFORCE にアクセスした日付を、以下の形式で表示します。

```
bruno <bruno@bruno_ws> (bruno) accessed 2005/03/07
dai <dai@p4demo.com> (Dai Sato) accessed 2005/03/04
earl <earl@p4demo.com> (Earl Ashby) accessed 2005/03/07
gale <gale@p4demo.com> (Gale Beal) accessed 2001/06/03
hera <hera@p4demo.com> (Hera Otis) accessed 2001/10/03
ines <ines@p4demo.com> (Ines Rios) accessed 2005/02/02
jack <jack@submariner> (jack) accessed 2005/03/02
mei <mei@p4demo.com> (Mei Chang) accessed 2001/11/14
ona <ona@p4demo.com> (Ona Birch) accessed 2001/10/23
quinn <quinn@p4demo.com> (Quinn Cass) accessed 2005/01/27
raj <raj@p4demo.com> (Raj Bai) accessed 2001/07/28
vera <vera@p4demo.com> (Vera Cullen) accessed 2005/01/15
```

ワークスペースの表示

クライアント・ワークスペースの情報を表示するには、p4 clients コマンドを発行します。このコマンドは、クライアント・ワークスペース名、ワークスペースの最終更新日付、ワークスペース・ルート、およびワークスペースの説明を以下の形式で表示します。

```
Client bruno_ws 2005/03/07 root c:\bruno_ws ''
Client dai-beos-locust 2002/10/03 root /boot/home/src ''
Client earl-beos-aspen 2002/04/15 root /boot/src ''
Client earl-dev-beech 2002/10/26 root /home/earl ''
Client earl-dev-guava 2002/09/08 root /usr/earl/develoment ''
Client earl-dev-yew 2004/11/19 root /tmp ''
Client earl-mac-alder 2002/03/19 root Macintosh HD:earl ''
Client earl-os2-buckeye 2002/03/21 root c:\src ''
Client earl-qnx-elm 2001/01/17 root /src ''
Client earl-tupelo 2001/01/05 root /usr/earl ''
```

ディポを一覧表示する

ディポを一覧表示するには、p4 depots コマンドを発行します。このコマンドによる表示項目は、ディポ名、ディポの作成日、ディポのタイプ（local、remote または spec）、ディポのホスト名または IP アドレス（remote の場合）、ローカル・ディポへのマッピング、およびディポに関するシステム管理者のコメントです。

単一の PERFORCE サーバで複数のディポを定義する方法について詳しくは、*『PERFORCE システム管理者ガイド』* をご覧ください。

サンプル・スクリプト

以下のサンプル・スクリプトでは、p4 fstat コマンドの出力を解析し、最新リビジョンがクライアント・ワークスペースにないときに作業状態にされているファイル（問題発生の可能性あり）をレポートします。

例: p4 fstat *コマンドの出力の解析結果を示すサンプル・シェル・スクリプト*

```
#!/bin/sh
# Usage: opened-not-head.sh files
# Displays files that are open when the head revision is not
# on the client workspace
echo=echo
exit=exit
p4=p4
sed=sed

if [ $# -ne 1 ]
then
     $echo "Usage: $0 files"
     $exit 1
fi

$p4 fstat -Ro $1 | while read line
do
     name=`$echo $line | $sed 's/^[\. ]\+\([^ ]\+\) .*$/\1/'`
     value=`$echo $line | $sed 's/^[\. ]\+[^ ]\+ \(.*\)$/\1/'`

     if [ "$name" = "depotFile" ]
     then
         depotFile=$value

     elif [ "$name" = "headRev" ]
     then
         headRev=$value

     elif [ "$name" = "haveRev" ]
     then
         haveRev=$value

         if [ $headRev != $haveRev ]
         then
          $echo $depotFile

         fi

     fi
done
```

付録 A

用語集

用語	定義
2 ウェイ・マージ	2 つのファイル・リビジョンを組み合わせるプロセス。2 ウェイ・マージでは、ファイル間の差異は表示できますが、衝突は表示できません。
3 ウェイ・マージ	3 つのファイル・リビジョンを組み合わせるプロセス。3 ウェイ・マージによって、どこで衝突の原因となる変更がなされたかを知ることができ、衝突の解決方法を指定できます。
base	衝突する 2 つの新しいファイル・リビジョンの元となったファイル・リビジョン。
binary ファイルタイプ	非テキスト・ファイルに割り当てられた PERFORCE ファイルタイプ。デフォルトでは、各リビジョンの内容は完全な形で保存され、ファイルそのものは圧縮フォーマットで保存されます。
list アクセス権限	レポート・コマンドの実行を許可し、ファイル内容へのアクセスを禁止するプロテクション・レベル。
MD5 チェックサム	アーカイブされたファイルの整合性を検証するために PERFORCE が使用する方法。
p4	PERFORCE のコマンドライン・クライアント・プログラムで、OS のコマンドラインから PERFORCE コマンドを実行するために実行するコマンド。
p4d	PERFORCE サーバ上にある、ディポとメタデータを管理するプログラム。
P4Diff	2 つのファイル間の差異を表示する PERFORCE アプリケーション。P4Diff は、ファイル衝突解決プロセスでファイルの比較に使用されるデフォルトのアプリケーションです。
P4Win	PERFORCE の Windows クライアント。Windows のエクスプローラ式のアプリケーションで、PERFORCE 操作やその結果の表示をグラフィカルに行えます。
PERFORCE サーバ	中央ホスト上の PERFORCE ディポとメタデータ。ディポとメタデータを管理するプログラムのことも指します。
RCS 形式	リビジョン管理システム形式。テキスト・ファイルのリビジョンの保存に使用されます。RCS 形式は、リバースデルタ・エンコーディングをファイルの保存に使用します。PERFORCE は、RCS 形式をテキスト・ファイルの保存に使用します。*リバースデルタ保存*もご覧ください。
read アクセス権限	PERFORCE が管理するファイルの内容を読み取ることを許可するプロテクション・レベル。
review アクセス権限	読み取り権限（read）とリスト・アクセス権限（list）を含む特別なプロテクション・レベルで、レビュー・コマンド（review）を実行するパーミッションを与えます。

用語	定義
super アクセス権限	*すべての* PERFORCE コマンドを実行する権限をユーザに与えるアクセス・レベル。対象となるコマンドには、プロテクションの設定、トリガのインストールおよび保守のためのサーバ停止を実行するコマンドが含まれます。
symlink ファイルタイプ	UNIX のシンボリック・リンクに割り当てられた PERFORCE のファイルタイプ。非 UNIX クライアントでは、symlink ファイルはテキスト・ファイルとして保存されます。
text ファイルタイプ	ASCII テキストのみを含むファイルに割り当てられた PERFORCE のファイルタイプ。*binary ファイルタイプ*もご覧ください。
theirs	ファイル衝突を解決する際、クライアント・ファイルとマージされるディポ内のリビジョン。ブランチ作成されたファイルで作業している場合、theirs は反映元ファイルのことです。
yours	ファイル衝突を解決するとき、クライアント・ワークスペース内の編集済バージョンのファイル。ブランチ作成されたファイルを反映させる際の反映先ファイルのことも指します。
アクセスレベル	ユーザが実行できる PERFORCE コマンドを制御するためにユーザに割り当てられる権限。*プロテクション*をご覧ください。
アップル・ファイルタイプ	AppleSingle フォーマットで保存される Macintosh ファイルに割り当てられた PERFORCE ファイルタイプ。データ・フォークとリソース・フォークを単一のファイルとして保存できます。
アトミック・チェンジ・トランザクション	1 つのトランザクションに多くのファイルへの操作をグループ化すること。トランザクションですべてのファイル操作が正常に実行された場合、すべてのファイルが更新されます。トランザクションでファイル操作が 1 つでも失敗すると、どのファイルも更新されません。
インター・ファイル・ブランチ機能	PERFORCE 専用のブランチ・メカニズム。
カウンタ	PERFORCE が使用する数値変数で、レビュー機能でチェンジリスト番号を追跡するために使用されます。
書き込み権限	ディポ内のファイルの内容を変更するコマンドの実行を許可するプロテクション・レベル。書き込み権限（`write`）には、読み取り権限（`read`）とリスト・アクセス権限（`list`）が含まれます。
管理者アクセス権限	*メタデータ*をオーバライドする PERFORCE コマンドを実行するために、ユーザに割り当てられるアクセス権限。ただし、サーバの状態を変更することはできません。
クライアント・サイド	クライアント・ビューにおけるマッピングの右側。対応するディポ・ファイルがクライアント・ワークスペースのどこにあるかを指定します。
クライアント・フォーム	クライアント・ワークスペースの定義に使用される PERFORCE フォーム。
クライアント名	現在のクライアント・ワークスペースを一意に示す名前。
クライアント・ルート	クライアント・ワークスペースのルート・ディレクトリ。1 つのマシンに 2 つ以上のクライアント・ワークスペースがある場合、それらは同じルート・ディレクトリを共有できません。
クライアント・ワークスペース	クライアント・マシンにおいて PERFORCE が管理するファイル・リビジョンの作業を行うディレクトリ。このクライアント・ワークスペースの名前は、デフォルトで、クライアント・ワークスペースがあるホストの名前に設定されています。デフォルトの名前を変更するには、環境変数 `P4CLIENT` を設定します。クライアント・ワークスペース、ラベル、およびブランチ仕様に同じ名前を付けることはできません。

用語	定義
クライアント・ワークスペース・ビュー	ディポ内のファイルの場所とクライアント・ワークスペースの対応を規定するマッピングのセット。
グループ	PERFORCE ユーザのリスト。
コードライン	まとめて進化するファイル・セット。コードラインは他のコードラインからブランチ作成することができ、これにより各ファイル・セットを個別に進化させることができます。
更新時刻	ファイルが最後に変更された時刻。
サーバ	PERFORCE において、クライアント・プログラムから送られたコマンドを実行するプログラム。PERFORCE サーバ（P4D）は、ディポ・ファイルやファイルについての記述を含むメタデータを保持し、クライアント・ワークスペースの状態を追跡します。
サーバ・ルート	サーバ・プログラムが自己のメタデータとすべての共有ファイルを保存するディレクトリ。サーバ・ルートを指定するには、環境変数P4ROOTを設定します。
最新リビジョン	ディポ内のファイルの最も新しいリビジョン。ファイル・リビジョンは順番に番号付けされるので、このリビジョンはそのファイルで一番大きな番号の付いたリビジョンとなります。
作業状態のファイル	クライアント・ワークスペースで変更中のファイル。
作業中チェンジリスト	まだサブミットされていないチェンジリスト。
削除ファイル	PERFORCE において、最新リビジョンに削除マークが付いたファイル。そのファイルの以前のリビジョンは、依然として利用できます。
サブスクライブ	特定のファイルに作用するチェンジリストがサブミットされたときに電子メールを受け取るよう登録すること。
サブミット	作業中チェンジリストと変更済ファイルを PERFORCE サーバに送ること。
差分	(*名詞*) 2 つのファイルを比較したときに一致しない行セット。ファイル*衝突*は 2 つのファイルそれぞれと、共通の第 3 のファイルとの差分が異なっていることです。 (*動詞*) ファイルの内容またはファイル・リビジョンを比較すること。
ジャーナル	最後のチェックポイント以降に PERFORCE サーバのメタデータに対して加えられたすべての変更の記録を含むファイル。
ジャーナル作成	PERFORCE サーバのメタデータに対して加えられた変更を記録するプロセス。
状態	チェンジリストについては、チェンジリストが新規なのか作業中なのかサブミットされたのかを示す値。ジョブについては、ジョブが作業中なのか終了したのか保留中なのかを示す値。ジョブの状態はカスタマイズできます。
衝突	衝突には 2 つのタイプがあり、第 1 のタイプの衝突は、2 人のユーザが1つのファイルを編集目的で作業状態にしたときに起こります。ユーザ A がファイルをサブミットした後は、衝突が起こるので、ユーザ B はサブミットできません。このタイプの衝突は、2 人のユーザが同じファイルを作業状態にしたことが原因となります。 第 2 のタイプの衝突は、ユーザがある 1 つのファイルを別のファイルにマージしようとするときに起こります。このタイプの衝突は、2 つのファイルと共通のベースとの比較結果が異なるとき、つまり、2 つのファイルの変更内容が異なるときに生じます。この場合、マージは自動では行われず、手動で行う必要があります。このタイプの衝突は、*差分*が一致しないことが原因となります。 *ファイル衝突*をご覧ください。

用語	定義
衝突解決	ファイルの 2 つのリビジョン間の差異を調整するのに使用するプロセス。
衝突再解決	ファイルの衝突を解決した後、サブミットする前に、そのファイルの衝突を再び解決するプロセス。
除外アクセス権限	指定したファイルへのアクセスを拒否する権限。
除外マッピング	特定ファイルを除外するビュー・マッピング。
ジョブ	PERFORCE が追跡するユーザ定義の作業ユニット。追跡する情報は、ジョブ・テンプレートで決定します。テンプレートは PERFORCE のシステム管理者が変更できます。
ジョブ仕様	PERFORCE ジョブとして保存されたフィールドと有効値を含む仕様。
ジョブの修正	チェンジリストにリンクされたジョブ。
ジョブ・ビュー	PERFORCE ジョブの検索に使用されるシンタックス。
所有者	特定のクライアント、ブランチ、あるいはラベルを作成した PERFORCE ユーザ。
所有リスト	現在クライアント・ワークスペースにあるファイル・リビジョンの一覧。
怠惰なコピー	ディポ内のファイル内容を複製せずに、ファイルの内部コピーを作成するのに PERFORCE が使用する方法。怠惰なコピーは、ファイルのコピーの代わりにオリジナル・ファイルへのリファレンスを保存することにより、使用するディスク容量を最小化します。
タイプマップ	ファイルにPERFORCEファイルタイプを割り当てるPERFORCEテーブル。
チェックポイント	ある特定の瞬間における基礎的なサーバ・メタデータのコピー。*メタデータ*をご覧ください。
チェンジリスト	PERFORCE におけるアトミック・チェンジ・トランザクション。チェンジリストで指定された変更は、チェンジリストがディポにサブミットされるまでディポには保存されません。
チェンジリスト番号	数値で表されるチェンジリストの一意の識別子。
チェンジリスト・フォーム	チェンジリストの修正に使用される PERFORCE フォーム。
チェンジ・レビュー	ディポ内の指定ファイルに加えられる変更を登録ユーザに電子メールで送信するプロセス。
チップ・リビジョン	PERFORCE では*最新リビジョン*のこと。*チップ・リビジョン*は、他の SCM システムで使用される用語です。
ディポ	PERFORCE サーバ上のファイル・リポジトリ。ここにはディポにサブミットされた全ファイルの全バージョンが含まれます。1 つのサーバに複数のディポを置くことができます。
ディポ・サイド	クライアント・ビューにおけるマッピングの左側。ディポ内におけるファイルの場所を指定します。
ディポ・シンタックス	ディポ内におけるファイルの場所を指定する PERFORCE シンタックス。
ディポ・ルート	ディポのルート・ディレクトリ。
デフォルト・チェンジリスト	番号付チェンジリストが指定されない限り、PERFORCE コマンドが使用するチェンジリスト。ファイルが編集目的で作業状態になると、デフォルトの作業中チェンジリストが自動的に作成されます。
デフォルト・ディポ	名前が指定されていないディポに与えられる名前。デフォルト・ディポ名は、`depot` です。
デルタ	2 つのファイル間の差異。
同期	ディポからクライアント・ワークスペースへ、ファイル・リビジョン（あるいはファイル・リビジョン・セット）をコピーすること。

用語	定義
トリガ	チェンジリストがサブミットされたとき PERFORCE サーバが自動的に起動するスクリプト。
反映	2 つのファイル・セット（例えば、2 つのコードライン・ブランチ）を比較すること。および ・ 一方のファイル・セットのどの変更を他方のセットに適用すべきかを判断する ・ それらの変更がすでに伝達されているかどうか判断する ・ 未処理の変更を伝達する こと。
反映先ファイル	ブランチ作成されたコードラインとオリジナルのコードライン間の変更を反映させる際、反映元ファイルから変更を受け取るファイル。
反映元ファイル	1 つのファイルから別のファイルへ変更を伝えるときの、変更の取り込み元ファイル。
番号付チェンジリスト	PERFORCE が番号を割り当てた作業中チェンジリスト。
非接続	PERFORCE サーバに接続できないクライアント・マシン。
非存在リビジョン	ファイルが完全に空のリビジョン。ファイルの非存在リビジョンに同期すると、ワークスペースからファイルが削除されます。ファイルの削除やリビジョン指定子 #none によって作成された空のファイル・リビジョンは、非存在ファイル・リビジョンの例です。
ビュー	2 つのファイル・セット間の関係を説明するもの。*クライアント・ワークスペース・ビュー*、*ラベル・ビュー*、*ブランチ・ビュー*もご覧ください。
ファイル衝突	3 ウェイ・マージにおいて、あるファイルの 2 つのリビジョンが互いに異なり、ベースファイルとも異なっている状況。 また、ディポ内のファイルの最新でないリビジョンを編集したファイルをサブミットしようと試みること。多くの場合、誰かがファイルを編集目的で作業状態にした後、別のユーザがそのファイルを編集目的で作業状態にしたときに起こります。
ファイルタイプ	ファイルの属性。PERFORCE では、ファイルを保存し差分表示する方法はファイルタイプで決まります。ファイルタイプの例は、`text`（テキスト）および`binary`（バイナリ）です。
ファイル・ツリー	ルート・ディレクトリ配下のすべてのサブディレクトリとファイル。
ファイル・パターン	PERFORCE のコマンドライン・シンタックスで、ワイルドカードを使用してファイルを指定できます。
ファイル・リビジョン	ディポ内のファイルの特定バージョン。各リビジョンには、順番に番号が割り当てられています。ディポ内にあるどのリビジョンにも、リビジョン番号（例えば、`testfile#3`）でアクセスできます。
ファイル・リポジトリ	全ユーザが共有する、すべてのファイルのマスター・コピー。PERFORCE では、*ディポ*と呼ばれます。
フォーム	一部の PERFORCE コマンドが表示する画面。例えば、PERFORCE チェンジ・フォームを使用して、特定のチェンジリストに関するコメントを入力し、そのチェンジリストに関連するファイルを検証することができます。
ブランチ	(*名詞*: ブランチ) オリジナルのファイルを追加することで作成されるコードラインに対して、あるコードラインをコピーすることで作成されるコードライン。*ブランチ*は*ブランチ・ビュー*の同義語としても使用されます。 (*動詞*: ブランチ作成) `p4 integrate` でコードライン・ブランチを作成 すること。

用語	定義
ブランチ仕様	オリジナルのコードラインとブランチの場所を定義することにより、ブランチの作成方法を規定します。ブランチ仕様は、ブランチの作成と更新のために反映プロセスで使用されます。クライアント・ワークスペース、ラベル、およびブランチ仕様に同じ名前を付けることはできません。
ブランチ・ビュー	ディポ内の 2 つのコードライン間のブランチ関係を示す仕様。各ブランチ・ビューは一意の名前を持ち、反映元のコードラインから反映先のコードラインにどのようにファイルをマッピングするかを定義します。*ブランチ*をご覧ください。
ブランチ・フォーム	ブランチ仕様の修正に使用する PERFORCE フォーム。
フル・ファイル保存	PERFORCE がディポ内にバイナリ・ファイルのリビジョンを保存する方法。すべてのファイル・リビジョンは完全な形で保存されます。これに対して*リバースデルタ保存*は、PERFORCE がテキストタイプのファイルに対して使用する方法です。
プロテクション	PERFORCE サーバのプロテクション・テーブルに格納されているパーミッション。
マージ	2 つの衝突しているファイル・リビジョンの内容を 1 つのファイルに結合するプロセス。
マージ・ファイル	2 つの衝突しているファイル・リビジョンから PERFORCE が生成したファイル。
マッピング	ビュー内の 1 行で、ディポ内のファイルとクライアント、ラベル、またはブランチ内のファイルとの対応を指定する左側と右側とで構成されています。左側はディポ・ファイルを指定し、右側がクライアント・ファイルを指定します。 (*クライアント・ワークスペース・ビュー*、*ラベル・ビュー*、*ブランチ・ビュー*もご覧ください。)
メタデータ	PERFORCE サーバが保存するデータで、ディポ内のファイル、クライアント・ワークスペースの現在の状態、プロテクション、ユーザ、クライアント、ラベル、およびブランチに関する記述があります。メタデータには、ファイルの実際の内容を除いて、サーバに保存されているすべてのデータが含まれています。
元に戻す	クライアント・ワークスペース内のファイルへの変更を廃棄すること。
ユーザ	誰が操作しているかを判断するため PERFORCE が使用する識別子。
ライセンス・ファイル	サイトの PERFORCE ユーザの数が、購入した数を超えないようにします。
ラベル	ユーザ指定のファイル・リビジョンのリストに名前を付けたもの。
ラベル・ビュー	ディポ内のどのファイル名が特定ラベル内に保存可能かを規定するビュー。
リソース・フォーク	Macintosh ファイルのフォークの 1 つ。(Macintosh ファイルは、リソース・フォークとデータ・フォークで構成されています。) PERFORCE のアップル・ファイルタイプを使用することにより、リソース・フォークを PERFORCE ディポ内に `AppleSingle` ファイルの一部として保存できます。
リバースデルタ保存	テキスト・ファイルのリビジョンを保存するために PERFORCE が使用する方法。PERFORCE は、最新リビジョンの全テキストを保存し、さらに、各リビジョンとその直前のリビジョンとの間の変更を保存します。
リビジョン指定	ファイル名に付く拡張子で、そのファイルの特定のリビジョンを指定します。リビジョン指定子として使用できるのは、リビジョン番号、チェンジ番号、ラベル名、日付 / 時刻の指定、あるいはクライアント名です。

用語	定義
リビジョン範囲	指定したファイルのリビジョン番号の範囲。範囲の最小と最大とで指定します。例えば、`myfile#5,7` はファイル `myfile` のリビジョン 5 から 7 を指定します。
リビジョン番号	ファイルのどのリビジョンを参照しているかを示す番号。
リモート・ディポ	現在 PERFORCE サーバではないサーバにあるディポ。
レビュー・デーモン	`p4 review` コマンドを使用するデーモン・プロセス。*チェンジ・レビュー*もご覧ください。
ローカル・シンタックス	ファイル名を指定する OS 固有のシンタックス。
ローカル・ディポ	現在の PERFORCE サーバにあるすべてのディポ。
ログ	PERFORCE サーバからのエラー出力。デフォルトでは、エラー出力は標準エラーに記述されます。ログ・ファイルを指定するには、`P4LOG` 環境変数を設定するか、サーバ起動時に `p4d -L` オプションを使用します。
ロック	PERFORCE のファイル・ロックは、ロックされたファイルを他のクライアントがサブミットするのを防ぎます。`Unlock` コマンドを実行するか、ロックされたファイルを含むチェンジリストをサブミットすることにより、ファイルのロックが解除されます。
ワークスペース	*クライアント・ワークスペース*の項をご覧ください。
ワイルドカード	文字列中の文字に適合させるために使用される特殊文字。PERFORCE のワイルドカードは次のとおりです。 ・ `*` はスラッシュ以外のすべての文字と適合する ・ `...` はスラッシュを含むすべての文字と適合する ・ `%%0` から `%%9` はビュー内のパラメータの置換に使用される

付録 B

PERFORCE のファイルタイプ

PERFORCE では一連のファイルタイプがサポートされており、ファイルタイプに従って PERFORCE サーバによるファイルの保存方法やファイル比較の可否が決定されます。ファイルを追加するとき、PERFORCE は自動的にファイルタイプを識別しようとします。最初に PERFORCE は、ファイルが通常のファイルかシンボリック・リンクかを識別し、次にファイルの最初の部分を調べてそれがテキストかバイナリかを判断します。非テキスト・キャラクタが見つかれば、そのファイルはバイナリとみなされ、見つからない場合はテキストとみなされます。(Unicode 環境ではファイルの検出方式が異なります。107 ページの「PERFORCE のファイルタイプ検出と Unicode」を参照してください。)

PERFORCE が制御しているファイルのタイプを識別するには、`p4 opened` コマンドまたは `p4 files` コマンドを発行します。PERFORCE ファイルタイプを変更するには、`-t filetype` オプションを指定します。ファイルタイプの変更について詳しくは、*『PERFORCE コマンド・リファレンス』* の `p4 add`、`p4 edit`、および `p4 reopen` の各項の説明を参照してください。

PERFORCE のファイルタイプ

PERFORCE は以下のファイルタイプをサポートします。

キーワード	記述	コメント	PERFORCE サーバによる記憶タイプ
`apple`	Macintosh ファイル	Mac データフォーク、リソース・フォーク、ファイルタイプおよびファイル・クリエータの AppleSingle 記憶。 詳細については、Mac クライアント・リリースノートを参照。	フル・ファイル、圧縮、AppleSingle フォーマット
`binary`	非テキスト・ファイル	ワークスペースにバイナリ・ファイルとして同期されます。ディポ内に圧縮された状態で記憶されます。	フル・ファイル、圧縮
`resource`	Macintosh リソース・フォーク	(旧式) このタイプは後方互換性を保つためにサポートされていますが、`apple` ファイルタイプを使用することを推奨します。	フル・ファイル、圧縮
`symlink`	シンボリック・リンク	UNIX および BeOS クライアントのマシンでは、これらのファイルをシンボリック・リンクとして処理します。非 UNIX クライアント・マシンはこれをテキスト・ファイルとして処理します。	デルタ
`text`	テキスト・ファイル	ワークスペースにテキストとして同期されます。行末変換が自動的に実行されます。	デルタ

キーワード	記述	コメント	PERFORCE サーバによる記憶タイプ
unicode	unicode ファイル	Unicode モードの PERFORCE サーバは unicode ファイルタイプをサポートします。このタイプのファイルは P4CHARSET により指定されたローカルの文字セットに変換されます。 Unicode モードで稼動していない PERFORCE サーバでは、Unicode ファイルタイプはサポートされていません。 詳細は、*『PERFORCE 国際語モードに関する注意事項』*をご参照ください。	UTF-8
utf16	unicode ファイル	PERFORCE サーバが Unicode モードで動作している場合、ファイルは P4CHARSET により指定されたローカルの文字セットに変換されます。 PERFORCE サーバが Unicode モードで動作していない場合、ファイルは UTF-8 で転送され、クライアント・ワークスペースにおいて UTF-16（BOM 付き、クライアント・マシンに適切なバイト順）にコード変換されます。 詳細については、「*PERFORCE 国際語モードに関する注意事項（i18nnotes.txt）*」をご覧ください。	UTF-8

ファイルタイプ修飾子

特定のファイルのベース・タイプにファイルタイプ修飾子を適用することにより、タイムスタンプの保存、RCS キーワード拡張、サーバ上でのファイルの保存方法の指定などができます。修飾子のファイルタイプへの適用に関して詳しくは、106 ページの「サーバでのファイルの保存形式を指定する」をご覧ください。

以下の表に、ファイルタイプ修飾子を示します。

修飾子	記述	コメント
+C	サーバが各ファイル・リビジョンのフル圧縮バージョンを記憶する	デフォルトのサーバの記憶メカニズム（バイナリ・ファイル用）
+D	サーバがデルタを RCS フォーマットで記憶する	デフォルトのサーバの記憶メカニズム（テキスト・ファイル用）
+F	サーバがリビジョンごとにフル・ファイルを記憶する	デルタの記憶が無意味または非効率的であるような、テキストとして処理されない長い ASCII ファイル（PostScript ファイルなど）に使用します。
+k	RCS（リビジョン制御システム）キーワード拡張	サポートしているキーワードは次のとおりです。 • $Author$ • $Change$ • $Date$ • $DateTime$ • $File$ • $Header$ • Id • $Revision$ RCS キーワードは大文字／小文字を区別します。 キーワードの後のコロン（例えば$Id:$）は任意です。

修飾子	記述	コメント
+ko	制限付きのキーワード拡張	キーワード Id および $Header$ だけを拡張します。この修飾子の対は、主に 2000.1 より前の PERFORCE にさか上って互換できるようにする目的で存在し、旧バージョンの PERFORCE における修飾子 +k (ktext) に対応します。
+l	排他的にファイルを作業状態にする（ロックする）	設定すると一度に 1 人のユーザしか、編集目的で作業状態にすることができません。 複数の作業者による変更のマージが不可能であるバイナリ・ファイルタイプ（グラフィック等）に有用です。
+m	オリジナルの更新日時を保存する	ローカル・ファイルシステム上のファイルのタイムスタンプがサブミット時に保存され、同期時に復元されます。Windows OS はファイルのタイムスタンプに依存するため、Windows 環境下におけるサードパーティの dll にとって有用です。デフォルトでは、更新日時はファイルを同期させた日時に設定されます。
+S	最新リビジョンだけがサーバで記憶される	旧リビジョンは、新しいリビジョンのサブミットによってディポから削除されます。実行ファイルまたは .obj ファイルに有用です。
+S*n*	最新の *n* 個のリビジョンだけがサーバに格納されます。*n* は 1 から 10 まで、または 16、32、64、128、256、512 のいずれかの数値です。	新しいリビジョンを *n* 個を超えてサブミットするか、既存の *+Sn* ファイルの *n* を現在のリビジョン数より少ない値に変更すると、古いリビジョンはディポから消去されます。
+w	ファイルは常にクライアント側で書き込み可能	PERFORCE によりファイルの読み取り／書き込みの設定が管理されるため、この設定は推奨されません。
+x	クライアント側で実行ビットをセット	実行ファイルに使用します。

サーバでのファイルの保存形式を指定する

通常、バイナリ・ファイルのファイル・リビジョンはすべてディポ内に保存されますが、テキスト・ファイルでは前のリビジョンが保存された後に加えられた変更だけが保存されます。これは*デルタ記憶*と呼ばれ、PERFORCE では RCS フォーマットを使用してそのデルタを記憶します。ファイルタイプにより、*ファイル全体を記憶*するか*デルタ記憶*を使用するかが決定されます。

ファイルタイプのいくつかは、ディポに記憶される際、gzip フォーマットに圧縮されます。圧縮はファイルをサブミットするときに行われ、解凍はファイルを同期（ファイルをサーバからワークスペースにコピー）させるときに行われます。クライアント・ワークスペースは常に、サブミットされた時点のファイルを保持します。

> **警告** *不用意にファイルが切り捨てられないようにするため、バイナリ・ファイルをテキストとして保存しないでください。Windows クライアント・マシンのバイナリ・ファイルをテキストとして保存すると、そのファイルに Windows のファイル終了キャラクタ ^Z が含まれている場合、^Z までの部分のみがディポに保存されます。*

Unicode のファイルにファイルタイプを割り当てる

PERFORCE サーバは、Unicode モードで起動して Unicode 文字を含むファイル名や PERFORCE メタデータに対するサポートを有効にするか、あるいは非 Unicode モードで起動することができます。非 Unicode モードではファイル名およびメタデータは ASCII でなくてはなりませんが、Unicode 文字を含むテキスト形式ファイルもサポートされます。

Unicode 文字を含むテキスト形式ファイルを管理する必要があるものの、PERFORCE メタデータ内に Unicode 文字が必要でない場合は、サーバを Unicode モードで起動する必要はありません。Unicode 文字を含むテキスト形式ファイルには、`utf16` というファイルタイプを割り当ててください。

サーバがどちらのモードを使用しているかは、システム管理者にお尋ねください

どちらのモードでも、PERFORCE では一連のファイルタイプがサポートされており、それによって PERFORCE サーバへのファイルの格納方法およびファイルの比較の可否が決定されます。以降のセクションでは、Unicode 環境においてテキスト形式ファイルを管理する際の考慮事項について説明します。

ディポにファイルを追加するときにファイルタイプを割り当てるには、`-t` オプションを指定します。例：

```
p4 add -t utf16 newfile.txt
```

ディポ内のファイルのファイルタイプを変更するには、`-t` オプションを指定してファイルを編集目的で作業状態にします。例：

```
p4 edit -t utf16 myfile.txt
```

ファイルタイプを選択する

Unicode を含むテキスト形式ファイルにファイルタイプを割り当てる際には以下のことを考慮します。

- ファイルの編集および比較が必要ですか？

  多くの IDE では、ユーザが手動で編集や比較を実行できない構成ファイルが作成されます。それらのファイルが変換されないように、当該ファイルには `binary` ファイルタイプを割り当てます。

- サイトでは異なる文字セットを使用するファイルを管理していますか？

  これに当てはまる場合は、当該ファイルを utf16 ファイルタイプを使用して格納することにより、変換は不可であるが比較が可能であるようにしてください。

Unicode モードのサーバでは、Unicode ファイルの内容を P4CHARSET で指定された文字セットに変換します。以下の表に、Unicode モードのサーバが様々な種類のテキストファイルをどのように管理するかについて、さらに詳しく示します。

テキストファイルのタイプ	サーバでの格納形式 (unicode モード)	検証の実施	P4CHARSET 単位で変換	クライアントのプラットフォーム単位で変換
text	拡張 ASCII	検証しない	変換しない	変換しない
unicode	UTF-8	検証する (UTF-16 および P4CHARSET として)	変換する	変換しない
utf16	UTF-8	検証する (UTF-16 として)	変換しない	変換しない

非 Unicode モードのサーバでは、unicode ファイルの内容について変換も検証も行いません。UTF-8 のデータは、クライアント・プラットフォームに適切なバイト順を使用して UTF-16 に変換されます。編集の際にそのようなファイルが破壊されないようにするため、編集用ソフトウェアでファイルを UTF-8 または UTF-16 として保存します。

テキストファイルのタイプ	サーバでの格納形式 (unicode モード)	検証の実施	P4CHARSET 単位で変換	クライアントのプラットフォーム単位で変換
text	拡張 ASCII	検証しない	変換しない	変換しない
unicode	UTF-8	検証する (UTF-16 および P4CHARSET として)	変換しない	変換しない
utf16	UTF-8	検証する (UTF-16 として)	変換しない	変換する

PERFORCE のファイルタイプ検出と Unicode

Unicode モードおよび非 Unicode モードの両方において、ディポにファイルを追加する際にファイルタイプを割り当てていない場合、PERFORCE はファイルの最初の 8192 文字を調べることによりファイルタイプの検出を試みます。印刷不能文字が検出されると、ファイルには binary のファイルタイプが割り当てられます。(Unicode モードではもう 1 つの検証が実行されます。印刷不能文字がなく、P4CHARSET により指定された文字セットを使用して変換可能な High ASCII 文字がある場合、ファイルには unicode ファイルタイプが割り当てられます。)

最後に (Unicode モードまたは非 Unicode モードで稼動しているサーバに対して)、UTF-16 BOM が存在する場合、ファイルには utf16 ファイルタイプが割り当てられます。それ以外の場合には、ファイルには text ファイルタイプが割り当てられます。(Unicode モードではもう 1 つの検証が実行されます。P4CHARSET によって指定された文字セットに定義されていない High ASCII 文字を含むファイルには binary ファイルタイプが割り当てられます。)

ほとんどの場合、PERFORCE により検出されたデフォルトのファイルタイプをオーバーライドする必要はありません。PERFORCE 検出のデフォルト・ファイルタイプをオーバーライドしなければならない場合は、p4 typemap コマンドを実行することにより、ファイルの拡張子に従って PERFORCE ファイルタイプを割り当てることができます。typemap 機能の使用に関して詳しくは、*『PERFORCE システム管理者ガイド』*および *『PERFORCE コマンド・リファレンス』*を参照してください。

ファイルタイプのオーバーライド

一部のファイル・フォーマット（Adobe PDF、リッチ・テキスト・フォーマットなど）は、実際にはバイナリであるのに、PERFORCE により誤ってテキストとして検出されることがあります。この問題を防止するため、システム管理者は `p4 typemap` コマンドを使用して特定のファイルタイプの保存方法を指定することができます。タイプマップ・テーブルに指定されたファイルタイプは、`-t filetype` オプションを付けて指定し直し、オーバーライドすることができます。

タイムスタンプの保存

通常、PERFORCE はファイルを同期させるときにタイムスタンプを更新します。更新日時（+m）修飾子は、ファイルの元のタイムスタンプを保存しておく必要がある開発者のために用意されています。この修飾子により、自分のクライアント・ワークスペースに同期させたファイルのタイムスタンプを、ファイルがサブミットされた時点のクライアント・マシンの時刻とすることができます。

Windows では、（開発環境下でも OS によっても）バージョン情報取得のためにサードパーティの `dll` のタイムスタンプが使用されます。+m 修飾子を使用するとオリジナルのタイムスタンプを保存できるため、誤ってバージョンが不一致となることを防ぐことができます。+m 修飾子は、（それが適用されるファイルに対する）クライアント・ワークスペースの `[no]modtime` の設定をオーバーライドします。この設定について詳しくは、104 ページの「ファイルタイプ修飾子」を参照してください。

RCS キーワード拡張

RCS（リビジョン管理システム）は初期のバージョン管理システムであり、ソース・ファイルに組み込み可能なキーワードが定義されています。RCS キーワードはファイルが格納場所に書き込まれるたびに更新されます。PERFORCE ではいくつかの RCS キーワードがサポートされています。1 つのファイルについて RCS キーワード拡張を有効にするには、+k 修飾子を使用します。RCS キーワードは次のように拡張されます。

キーワード	拡張対象	例
$Author$	ファイルをサブミットする PERFORCE ユーザ	$Author: bruno $
$Change$	サブミットされるファイルに付ける PRFORCE チェンジリスト番号	$Change: 439 $
$Date$	フォーマット *YYYY/MM/DD* による、最終サブミットの日付	$Date: 2000/08/18 $
$DateTime$	フォーマット *YYYY/MM/DDhh:mm:ss* による、最終サブミットの日付と時刻。 日付と時刻は、サブミット時点の PERFORCE サーバ上の現地時間に準ずる。	$DateTime: 2000/08/18 23:17:02 $
$File$	ディポ・シンタックスによるファイル名のみ（リビジョン番号なし）	$File: //depot/path/file.txt $
$Header$	Id と同義	$Header: //depot/path/file.txt#3 $
Id	ディポ・シンタックスによるファイル名およびリビジョン番号	$Id: //depot/path/file.txt#3 $
$Revision$	PERFORCE リビジョン番号	$Revision: #3 $

索引

こ

さ

し

す

せ

た

ち

る

ろ

わ