Magitユーザーマニュアル

Magit は、バージョン管理システムGitリポジトリへのインターフェースで、Emacsへの拡張機能として実装されています。Magitは、GNU Emacsのバージョン22以降をサポートしています。それは、他のemacs達も動作するかもしれませんが、Magit開発者は、サポートされていないバージョンで表示されたバグを調査し、修正するつもりはありません。でも、他のemacs達のバグ修正パッチや、互換性を維持するボランティア活動は歓迎します。

---

1 はじめに

Magit を使用すると、EmacsでGitリポジトリを検査および変更することができます。あなたは監視対象のファイルに加えた変更を確認し、コミットすることができ、たとえば、あなたは過去の変更履歴を閲覧することができます。Magitは、revertやmergeやrebaseや、その他一般的なGit操作のつまみ食いです。
MagitはGitへのインタフェース完全には網羅していません。一般的なGitの操作を便利にすることだけを目指しています。したがって、Magitを使えばGitそのものを知らなくて良くなる訳ではありません。
このマニュアルでは、すべてのMagit機能のツアーを提供しています。しかし、一般的なバージョン管理の紹介やGitの詳細な紹介ではありません。
Magitへの入り口は、M-x magit-statusで、Magitのステータスバッファにあなたを導きます。頻繁にそれを使用することになるので、magit-statusをお好みのキーにバインドするのはおそらくいい考えです。
ス テータスバッファに加えて、Magitはコミットをリストを表示するバッファ、差分のバッファ、および他の種類のバッファも作成します。これらのバッファの全てがmagit-mode内にあり、同じキーバインディングを持っています。コマンドの全てが全コンテキストで使える訳ではありませんが、全てのMagitバッファにおいて、入力したキー

は常に同じ動作をします。
当然のことながら、Magitは多くの作業をするためにgitコマンドを走らせます。*magit-process*バッファには、最後に走らせたコマンドの実行結果があります。あなたは$でそのバッファに切り替えることができます。

---

2 謝辞

Marius Vollmer がプロジェクトを始めました。ありがとうございます！
最初のMagit発表の初日から、John Wiegleyは多数の修正、UIの改善、および新機能に貢献しています。ありがとうございます！
Linh Dangと Christian Neukirchenも初日から貢献した。ありがとうございます！
Phil Hagelbergは数日後に加わりました。ありがとうございます！
Alex Ottはgit svnのためのサポートに貢献してくれました。ありがとうございます！
Marcin Bachry はバグ修正や装飾されたログのサポートに貢献しました。ありがとうございます！
Alexey Voinovはバグを修正しました。ありがとうございます！
Rémi Vanicat はtrampのサポートを手伝ってくれました。ありがとうございます！

---

3 セクション

すべてのMagitバッファはネストされた 'セクション'に構造化されます。これらのセクションは非表示にすることができ、そして個別に表示することができます。 セクションを非表示にしたときは、最初の行のみ表示されていて、その子の全ては完全に隠されています。
セクションの可視性を制御するための最もきめの細かい方法は、Tabキーです。TABキーによって現在のセクション(ポイントを含むセクション)は非表示と表示を切り替えます。
S-TABをタイプすると、現在のセクションの子の、表示と非表示を切り替えます。それらのすべてが表示されている場合、それらはすべて非表示になります。一方、一部が隠されている場合、それらのすべてを表示します。
数字キー1、2、3、4はセクションの可視性レベルを制御します。例えば、2を打つと、レベル1および2のセクションが表示され、レベル3のセクションを非表示にします。しかし、現在のセクションの親または子だけのセクションが影響を受けます。
たとえば、現在のセクションは、レベル3にあり、あなたが1をタイプしたとき、現在のセクションの祖父（レベル1にある）が表示され、そして、現在のセクション（レベル2）の親が非表示になります。他のセクションの可視性は変更されません。
これは、少々複雑に聞こえるかもしれませんが、じきに把握できるでしょう。
M-1、M-2、M-3、M-4は1、2、3、4に似ていますが、現在のセクションに関連しているかどうかにかかわらず、今度はそれぞれのレベルのすべてのセクションが影響を受けます。
たとえば、M-1は 、トップレベルのセクションの最初の行が表示されますし、それ以外をすべて非表示になります。一方、M-4を入力すると、すべてのものが表示されます。
ス テータスバッファの設定により、セクションの可視性の変更のいくつかは、他のものよりもより一般的である。レベル2はファイル単位で、レベル4はハンク単位です。したがって、あなたは、現在のファイルのdiffを折り畳むに2を入力することができ、そして、す べてのファイルを折り畳むにはM-2をタイプします。これは、修正したファイルにドリルダウンした後に、それをきちんと整え、デフォルトの構成にステータ スバッファを戻すための迅速な方法です。
2及びM-2はステータスバッファでよく使うので、より覚えやすいキーバインドが追加されています： M-h(非表示）とM-H(すべて非表示）です。 同様に4とM-4はM-s（表示)とM-S(すべて表示）として使用することもできます。
ステータスバッファ以外のバッファで、M-h, M-H, M-s, M-Sは、2と4の異なるレベルで動作するかもしれない しかし、彼らは彼らの一般的な意味を保つ： M-Hは 、すべての詳細を隠蔽し、そしてM-Sは、すべてを表示します。

4ステータス

M-x magit-statusを実行すると、Magitのメインインターフェイスであるステータスバッファを表示します。あなたは、それぞれが独自のGitリポジトリに関連付けられている、同時にアクティブな複数のステータスバッファを持つ事ができます。
(訳 注:emacsのカレントディレクトリをGitリポジトリのフォルダにして)Gitリポジトリ内でM-x magit-statusを呼び出すと、そのリポジトリ為のステータスバッファに切り替わります。それ以外の場合は、ディレクトリの入力を求めます。接頭引数を付けると、常にディレクトリの入力を求められます。
あ なたはmagit-statusがどのリポジトリにの為に動作するかを尋ねる方法をカスタマイズするためにmagit-repo-dirsをセットするこ とができます。magit-repo-dirs がnilの時、magit-statusは単にディレクトリを尋ねます。
Gitリポジトリでないディレクトリを指定した場合、M-x magit-statusはGitリポジトリとして初期化することを提案します。
magit- repo-dirsがnilではない時、それはディレクトリ名のリストとして扱われます。そしてmagit-statusは 、これらのディレクトリ内のすべてのGitリポジトリを見つけて、補完の為にそれらを提示します。（けれども、Magitはmagit-repo-dirs-depthで指定したレベルの深さまでしか探しません。）
2接頭引数を付けると、magit-status 、常に生のディレクトリを求めるプロンプトが表示されます。
し たがって、通常、あなたは多くのGitリポジトリを保持する場所をmagit-repo-dirsにセットして、そしてC-u M-x magit-statusでそれらの間を切り替えます。あなたが通常の作業領域外のリポジトリに移動したい場合は、または新しいリポジトリを作成する場合は、あなたはC-u C-u M-x magit-statusを使用します。
あ なたがEmacsの外でリポジトリに変更を加えたときは、明示的にステータスバッファをリフレッシュする必要があります。あなたは、ステータスバッファそれ自体にgとタイプすることができ、それはM-x magit-statusで、バッファを切り替えるときにC-x bとタイプする代わりに、使います。また、Emacsでファイルを保存した後も、このようにステータスバッファをリフレッシュする必要があります。
ス テータスバッファの先頭にあるヘッダーは、リポジトリの状態の要約を示しています。それがどこにあるか、どのブランチがチェックアウトされているか等です。ヘッダーの下に作業ツリーとステージング領域の詳細を示すいくつかのセクションがあります。前節で説明した通り、あなたはそれらを隠したり表示したりでき ます。
*Untracked files(*追跡されていないファイル*)*が存在した場合、最初のセクションではそれらを示しています。詳細はUntracked files(追跡されていないファイル)を見てください。
次の2つのセクションでは、ローカルの変更を示しています。次の章、ステージングおよびコミット でそれらを全部説明します。
現在のブランチがリモート追跡ブランチに関連付けられている場合は、ステータス·バッファは、現在のブランチと追跡ブランチの違いを示しています。詳しくはpushとpullを参照ください。
歴史の書き換えはセッション中、ステータス·バッファは、保留中の変更を示しており、保留中のセクションをコミットします。詳細はrewrite(書き換え)を参照ください。

---

5 Untracked files(追跡されていないファイル)

追跡されていないファイルは*Untracked files*セクションに表示されます。
あなたはsでステージング領域に、追跡されていないファイルを追加することができます。あなたがsとタイプしたときのポイントが、*Untracked files*セクションのタイトルにある場合は、すべての追跡されていないファイルがステージングされます。
C-u sとタイプすると、監視対象のファイルに対するすべての変更と一緒に、すべての追跡されていないファイルをステージングします。
あ なたはiとタイプして 、それらを無視するようにGitに指示することができます。これは、.gitignoreファイルにそのファイル名を追加します。C-u iとタイプすると無視したいファイルの名前を入力するように求められます。これは、ディレクトリ全体を無視するのに便利ですIコマンドは iと似ていますが、代わりに.git/info/excludeにファイルを追加します。
追跡されていないファイルを永遠に削除するにはkを使います。kをタイプしたとき、ポイントが*Untracked files*のセクションのタイトルにある場合は、追跡されていないファイルを全て削除します。

---

6 ステージとコミット

Gitのコミットは、2ステップのプロセスです。最初に、あなたが "ステージング領域"にコミットしたい変更内容を追加し、そして、あなたはそれをリポジトリにコミットします。これはあなただけのローカルな変更のサブセットをコミットすることを許します。
あなたが希望すれば、Magitはステージング領域を無視することができます。ステージング領域を使用しない時は、Magitは*Changes*という名のセクションに、あなたのコミットしていない変更を表示します。
ステージング領域が使用中であるときは、Magitは2つのセクションを使います。*Unstaged Changes*と*Staged changes*です。*Staged changes*セクションには、次のコミットに含まれる変更を示していて、*Unstaged changes*セクションでは、取り残される変更を示している。
ス テージング領域にコミット予定のハンクを上げるには、ポイントをハンクに移動して、sをタイプします。同様に、ハンクをステージから下ろすためには、そこにポイントを移動しuとタイプします。あなたはSまたはUを入力したときのポイントがdiffのヘッダ内にある場合は、その差分に属するすべてのハンクを同時に上げ下げします。
s またはuとタイプしたときリージョン(選択領域)がアクティブであれば、そのリージョン(選択領域)内の変更だけをステージに上げたりステージからおろし たりします。(これは、行ごとに動作します。行の先頭がこのリージョン(選択領域)にある場合は変更に含まれていますが、それ以外の場合は動作しませ ん。)
ハンクのサイズは+または-のタイプで増減します。0タイプすると、ハンクのサイズをデフォルトに戻します。
C-u sのタイプでは、ステージに上げるファイルの名前を尋ねてきます。これは例えば、ステージに上げるファイルが隠れている時に使います。
全ての差分のハンクを一気にステージに上げるには、Sとタイプします。全てをステージから下ろすには、Uとタイプします。
C-u Sとタイプすると、追跡対象のファイルへの変更に加えて、すべての追跡されていないファイルをステージに上げます。
あなたはハンクにポイントを移動してkとタイプすることで、コミットされていない変更を破棄することができます。破棄する変更部分はsとuによって選択されたものです。
コミットする前に、変更内容の簡単な説明を記述すべきです。
あなたの変更内容を記述できるバッファをポップアップするにはcとタイプしてください。あなたが記述に満足したら、コミットを実行するために、そのバッファ内でC-c C-cとタイプしてください。
ChangeLogファイルに変更を書きたい場合、あなたは、diffのハンクにC-x 4を使用することができます。
ス テージング領域が使われないときにcとタイプするのは特殊な状況です。通常、その次のコミットには空になりますが、しかし、あなたはmagit-commit-all-when-nothing-staged変数をカスタマイズすることで、より有用な何かをMagit に設定することができます。一つの選択肢は、すべての変更をコミットするために、後続のC-c C-cを指示することです。もう一つの選択肢はcとタイプした時の全てをステージに上げることです。
あなたはC-c C-aとタイプして、変更内容の説明のバッファ内でHEADの現在のコミットの変更を次のコミットで変更するかどうかを決定するフラグをトグルすることができます。
C-c C-sは--signoffオプションをトグルします。デフォルトはmagit-commit-signoffカスタマイズ変数で決定されます。
C-c C-eとタイプすると、--allow-emptyオプションをトグルします。これで、あなたは変更を一切含まない、メモとしてのみ役立つコミットを行うことができます。
C-c C-tとタイプすると、コミットの作者の名前と電子メールアドレスを指定するためのオプションをトグルします。デフォルトはgitの構成設定のuser.nameとuser.emailにより決定されます。
あ なたが*magit-log-edit*バッファにいるとき、もしあなたが心変わりして、あなたのコミットを進めたくない場合、あなただけの、別のバッ ファに切り替えることができます そこで編集を続け、あなたが幸せになるまでそれをステージに上げたりステージから下ろしたりでき、そしてそれからC-x bをタイプするかMagitバッファで再度c押すことで*magit-log-edit*バッファに戻ります。
あなたが*magit-log-edit*バッファを消去して葬り去りたい場合、C-c C-kとタイプすることで可能です。
cとタイプしたときも変更の説明バッファをポップアップしますが、しかし、加えて、ポイントのある場所に変更内容のChangeLog形式のエントリを挿します。

---

7 履歴

あなたの現在のヘッドのリポジトリの履歴を表示するためには、llとタイプしてください。簡潔な形式で履歴が表示

されている新しいバッファが閲覧できます。各コミットメッセージの最初の段落は、コミット間の関係の表示の横に表示されます。
そ のリポジトリのブランチ間や任意の2点間の履歴を見るためには、l r lとタイプしてください。あなたが出発点と履歴範囲の終了点の参照を入力するプロンプトが表示されます。あなたはそれらを指定するためにオートコンプリートを使用することができます。マスタにない新機能開発のためのブランチ上のコミットを表示するのは遠隔履歴ログの表示を使用する典型的なケースで、l r l master RET new-feature RET とタイプします。例えばそこから、それらのコミットを詳しく調べたりつまみ食いしたりすることができます。
より徹底したフィルタリングは、そのポップアップに表示されているように、lとそれに続く一つ以上の引数をタイプすることによって行うことができます。例えば= g('grep')は、ログメッセージが特定の文字列/正規表現に一致するコミットだけに出力を制限します。
l L(またはl C-u Lとタイプすると 、より詳細な形式でログを表示します。
Magitはmagit-log-cutoff-lengthエントリのみを表示します。eの表示2回でたくさんのエントリを表示します。C-u e はすべてのエントリを表示し、数値前置引数を指定すると、eは指定した数のエントリを追加します。
あなたは、コミットにポイントを移動してから、様々なものがそれで起こることがあります。(コミットされたけどまだプッシュされていないセクションに見るような以下のコマンド群は、コミットのリストのに対して働きます。)
RETとタイプすると 現在のコミットに関する詳細な情報をポップアップ表示して、その新しいバッファにポイントを移動します。コミットバッファ参照。SPCまたはDELのタイピングも情報を表示しますが、連続入力した場合、その新しいバッファをスクロールアップまたはスクロールダウンさせます。
a とタイプすると、現在のブランチに現在のコミットを適用します。これは、あなたが他の幾つかのブランチの履歴を閲覧して、そこから幾つかの変更をつまみ食 い(cherry-pick)したいときに便利です。典型的な状況は、開発版のプログラムの幾つかのバグフィックスをリリースブランチへ適用する時です。つまみ食い変更が自動的にコミットされることはありませんので、あなたはその変更を明示的にコミットする必要があります。
Aとタイプすると現在のコミットをつまみ食いして、そしてコンフリクトが無い時は自動的にその変更のコミットも行ないます。
vとタイプすると現在のコミットを元に戻します。従って、それは逆に、そのコミットによって行われた変更を適用します。これは明白な間違いが判明した変更を元に戻すのに便利です。aと同様に、変更を明示的にコミットする必要があります。
C-wとタイプすると、現在のコミットのsha1をキルリング(kill ring)へコピーします。
=とタイプすると、現在のコミットからマークしたコミットへの差分を表示します。
.とタイプすることで現在のコミットをマークすることができます。現在のコミットが既にマークされているときは、.のタイプでそのマークを外します。マークしたコミットにポイントが無い時にマークを外すにはC-uを使用します。
=のような一部のコマンドでは、現在のコミットとマークされたコミットを暗黙の引数として使います。他のコマンドでは引数を要求するときにマークされたコミットをデフォルト値として指定します。

---

8 Reflog

あなたの*reflog*、つまり、リポジトリのヘッドに加えられた変更のローカル履歴を参照するにはl hとl Hを使用することができます。l hでHEADのreflogを見ている時はHのタイプでヘッドを尋ねてきます。
結果バッファはちょうどそのコミット履歴を表示するl lとl Lによって生成されたバッファのようなものです。

---

9 コミットバッファ

(多分履歴やログバッファで選択した)コミットを閲覧するときは "コミットバッファ"が表示され、そのコミットについての情報を表示し、対話することができます。
diffやハンク内にカーソルを置いてaとタイプすることで、あなたの作業コピーに同一のじパッチを適用することができます。必ずしも全体のコミットをつまみ食いしたくはないが、別のブランチから変更をコピーしたいときに便利です。
vと入力することで、追加されたすべての行を削除し、削除されたすべての行を追加して、パッチを逆に適用することができます。これは、バグを導入したと判った後で変更を削除する便利な方法です。
コ ミットメッセージが参照している、リポジトリ内の他のコミットはそれぞれユニークなハッシュ値をもっているの

で、コミットメッセージが参照する、リポジトリ内でそれぞれユニークなハッシュ値によって表される他の任意のコミットを見たいなら、ハッシュ値をクリックする かカーソルを移動してRETを押下することで、そのハッシュはハイライト表示され、参照されているコミットを訪れることができます。
コ ミット·バッファは、そこに表示されているコミットの履歴を維持管理します。そのコミットメッセージに参照されているコミットを訪れた後はC-c C-bとタイプすることでそのコミットメッセージに戻ることができます。履歴の中を進むにはC-c C-fとタイプします。またはバッファの下部にあるボタン[back]と[forward]を使います。

---

10 差分(diff)

Magitは普通 "統一された"形式で差分を表示します。
diff を示す任意のバッファで、差分表示箇所のどこかでeとタイプすることで、その差分を2ファイルバージョンのEdiffで表示することができます。diff は、マージする必要があるステータスバッファ内のファイルの場合は、インタラクティブなマージツールとしてEdiffを使用することができます。それ以外の場合は、Ediffは、単にファイルの2つのバージョンが表示されます。
作業ツリーから別のリビジョンへの変更を表示するには、dとタイプします。任意の二つのリビジョン間の変更を表示するには、Dとタイプします
作 業ツリーに変更を適用するために、diffの出力内でaを使用することができます。通常ポイントがファイルのdiffヘッダにある場合、そのファイルに対するすべての変更が適用され、そして、それがハンクであるときは、そのハンクだけに適用されます。選択領域がアクティブなときは、変更の適用はその選択領域のみです。
vとタイプすると、選択した変更の適用を取り消します。

---

11 タグ付け

t tとタイプすると軽量(lighweight)タグを作ります。t aとタイプすると注釈付き(annotated)タグを作ります。通常コミットメッセージを書き込む為の*magit-log-editバッファにあなた を導きますが、C-c C-cとタイプするとコミットメッセージの代わりにタグを作ります。これは*magit-log-edit*バッファに追加されるTagフィールドによっ て制御されます。それは編集することも可能です。

---

12 リセット

一度、ローカルリポジトリへのコミットを追加した後は、どのような方法でもそれ以上コミット変更することはできません。しかし、現在位置を以前のコミットにリセットして再スタートすることができます。
あなたが既に履歴を公開している場合、この書き換え方は混乱を招く可能性がありますので避けるべきです。けれども、あなたのローカルの履歴を書き換えるのは構いません、それは多くの場合(例えばvのタイプで)コミットを戻すことによってミスを修正するクリーナーです。
xのタイプはリビジョンを尋ね、そして現在位置を入力したリビジョンにリセットします。作業ツリーとステージング領域は変更しません。したがって、ステータスバッファのステージされた変更のセクションには、あなたのコミット履歴から削除された変更が表示されます。こうして歴史を書き換え、それを今始めて行ったかのようにその変更をコミットできます。
ポイントがコミットが記述された行にあるときxをタイプすると、そのコミットをリセットするリビジョンの初期値に設定します。よって、プッシュされてないコミットセクションのコミットの一つにポイントを移動してx RETとタイプしてそれを現在位置にリセットできます。
Xとタイプすると最も最近のコミットの状態に作業ツリーとステージング領域をリセットします。これは、あなたのローカルの変更を破棄しますので注意してください。
指定したコミットへと現在のヘッドと作業ツリーの両方をリセットしたい場合はxにプレフィックスを与えることで可能です。これは、まずその現在位置をリセットするためにプレフィックス無しのxをタイプしてから、Xを使うのと同じです。

13 スタッシュ(stash)

あなたはZ Zで新しいスタッシュ(隠れ家)を作成することができます。スタッシュはステータスバッファに表示され、aとタイプしてそれらを適用することができ、そしてAとタイプしてそれらを開くことができます。スタッシュを削除するにはkとタイプします。
前置引数があると、aとAの両方とも、インデックスならびにスタッシュから作業ツリーを元に戻そうとします。
z -k -zとタイプすると、z zとタイプしたときと同様のなスタッシュを作成しますが、作業ツリーの変更とインデックスを削除します。これは、例えば、同じ変更の複数のバリエーションをテストするのを容易にします。
編集中に素早くスナップショットを作りたいだけの場合は、z sとタイプすると自動的にタイムスタンプをメッセージとして入力し、デフォルトの設定では作業ツリーとインデックスをそっくりそのまま維持します。
ス タッシュを訪れたり閲覧したりするのに便利な方法があります。SPCやDELのタイプでそのスタッシュの説明をポップアップ表示し、そしてスクロールし、RETのタイプでポップアップしたバッファにポイントを移動します。C-u RETを使用すると、他のウィンドウでそのバッファにポイントを移動します。

---

14 ブランチ

現 在のブランチがステータスバッファのヘッダに表示されます。b bとタイプして別のブランチに切り替えることができます。これは直ちにそのブランチをチェックアウトしワーキングコピーに展開しますので、ブランチを切り 替えている間はどんなローカルの変更もすべきではありません。
リモートブランチに切り替えようとした場合、Magitはそれに対応するローカルトラッキングブランチを作成することを提案します。 このため、簡単にリモートリポジトリに登場した新しいブランチでの作業を始めることができます。
ポイントがコミットの説明にあるときにb bとタイプすると、デフォルトに切り替えるようそれをコミットすることを提案します。これは、になりますヘッドが切り離さ。
b mとタイプするとブランチ名前を変更できるます。(同じ名前のブランチが既に明らかに存在しない限り)
新しいブランチを作成し、すぐにそれに切り替えるにはb nとタイプします。
ブランチを削除するにはb dとタイプします。削除するブランチ上で作業している場合、Magitは 'master'ブランチに切り替えることを提案します。
ブランチの削除はそれが既にHEADまたはの上流のブランチに完全にマージされている場合のみ可能です。b Dとタイプしない限り。ここに龍が...
b vとタイプして、そこで作業ができる*magit-branches*と呼ばれる新しいバッファにローカルおよびリモートブランチの一覧を表示します詳細についてはブランチマネージャを参照下さい。

---

15 ブランチマネージャ

ブランチマネージャは独自のローカルキーマップを有する*magit-branches*と呼ばれる独立したバッファです。
バッファは、ローカルとリモートの両方のブランチが含まれます。現在のローカルブランチは、名前の先頭に "*" がマークされています。
ブランチをチェックアウトするには、目的の枝にカーソルを移動してRETを押します。
kとタイプすると現在行のブランチを削除し、そして、現在のローカルブランチにマージされていない場合でも、C-u kでそれを削除します。削除はローカルとリモートの両方のブランチで動作します。
ローカルブランチ上でTを入力することによって、あなたはそれを追跡するために設定されているリモートブランチに変更できます。

---

16 どうなってんのー？

wとタイプすると、他のブランチが現在のブランチにどのように関係するかの概要を示します。
各ブランチでは、現在のブランチに含まれていない、そのブランチでコミットをリストするセクションを取得します。セクションは、最初は折りたたまれ、明示的コミットのリストを表示するにはTabキー（または類似の）でそれらを開く必要があります。
点が*N*でマージされていない上に...タイトルコミットである場合、対応するブランチはマージのデフォルトとして提供されます。
分岐タイトルにIを押すと、WAZZUPビューで、この分岐を無視します。あなたが無視も含めてすべての枝を、表示するようにC-u wを使用することができます。そのビューですでに無視枝に私を叩いても 、それをunignoreます。

---

17 マージ

Magitはブランチをマージする2つの方法を提供しています。手動および自動です。自動マージが直ちにコミットまで行ってしまう一方、手動マージは、作業ツリーとステージング領域にすべての変更を適用しますが、それらを自動ではコミットしません。
マージを開始するにはm mとタイプしてください。
マージを開始した後に、あなたはステータスバッファのヘッダをみて、次のコミットがマージコミット（複数の親を持つ）であるということを思い出すかもしれません。あなたが手動マージを途中で止めたい場合、xでHEADをハードリセットします。
あなたが2つのブランチ間で競合する変更をマージする場合は失敗する可能性があります。その場合、コミット前で自動マージは停止し、基本的に手動マージに後戻りします。あなたは、競合を例えばeで解決する必要があり、そして解決したファイルを例えばSでステージします。
あなたが解決した個々のハンクを一個づつステージすることはできません。それらのすべての競合が解決された後にのみ、ファイル全体をステージングできます。

---

18 リベース

ステータスバッファでRとタイプすると、リベースを開始したり、それがすでに進行中である場合、どのように継続するかを聞いてきます。
コ ンフリクトが原因でリベースが途中で停止すると、ステータスバッファのヘッダには現在選択中のコミットの流れからどのくらいへだたっているか表示されま す。こうなったときは、コンフリクトを解決し全てをステージしてからリベースを継続するためにR cとタイプします。あるいは、リベースの最中にcまたはCとタイプした時は、リベースを続行するかどうか問われます。
例えば、リベースの為にgit pullを設定する代わりにマージするようにさまざまな方法でリベースを始めることができます。このようなリベースもRとタイプすることで終えることができます。

---

19 書き換え

先 にほのめかしたように、コミット履歴を書き換えることができます。たとえばxとタイプすると先端位置を以前のコミットにリセットすることができます。 これは作業ツリーを変更しないままで、ステータスバッファには新しくセットした現在位置以降に行われたすべての変更が表示されます。 再び、それらの変更を複数のコミットに分割してコミットすることがでます。
直前のコミットを修正するのは、このようなよくある履歴書き換えの、特別なケースです。
履歴を書き換えるための別の一般的な方法は、以前のコミットに先端をリセットし、そしてそれから異なる順序で以前のコミットをつまみ食いすることです。例えば、reflogからそれらをつまむ事ができます。
Magitは書き換え関連で作業を簡素化する幾つかのコマンドを持っています。これらのコマンドは先頭の文字がrで始まります。
rbとタイプすると書き換え操作を開始します。ベースコミットの入力が求められます。現在位置までの、このコミット

およびそれ以降のすべてのコミットは*Pending commits*リストに入れられ、その後再び、現在位置はベースコミットの親にセットされます。これはmagit-rewrite-inclusive変数によって、git rebase、すなわちrewrite操作からベースコミットを除外するように振る舞うように設定することが出来ます。
そ の後、すべて適用済みになるまで、典型的なコマンドとしてaや{A/0}を使用して任意の順番でpending commitsのリストからつまみ食いしてコミットすることができます。Magitはコミットが適用済みであることを、マーカーを*から..に変更して示 します。
Aを使用すると、(いつものように)直ちにそのコミットをコミットします。以前の複数のコミットをまとめてひとつのコミットにしたいなら、aとタイプするとワーキングツリー全てを適用対象とし、それからそれらを一度にコミットします。
Magitは、マージコミット書き換えのための明示的なサポートを持っていません。上手に保留中のコミットのリストのマージコミットを取り込みますが、それらを自動的に再生する方法がありません。マージのやり直しは明示的行わなければなりません。
心変わりしたときはコミットを戻すのにvを使う事もできます。こうすると.マークが*に戻ります。
以前、ステータスバッファからしおり情報を削除するためにr sとタイプして書き換えを行ないました。
それを最初からやり直したい場合はr aとタイプします。これは書き換えを中断し、現在位置をr sで書き換えを始める前の値にリセットします。
r fとタイプすると、それら全てにAを使ったかのように、使ってないコミットを次々と適用し、書き換えを終了させます。
r *やr .を明示的に使って保留中のコミットの*や.マークを変更することができます。
ステータスバッファにはコミット保留中のリストに加えて保留中の変更が表示されます。そのセクションは、元の現在位置と今の現在位置の間の差分を示しています。依然として書き換えの必要がある変更を確認するためにそれを使用することができ、他のdiffからのように、それからのハンクを適用することができます。

---

20 プッシュとプル

P Pとタイプしたとき、Magitはgit pushを実行します。P Pに接頭引数を与えた場合は、プッシュするリポジトリの入力を求められます。現在のブランチで、まだ既定のリモートリポジトリが設定されてないときは、同 様にプッシュするリポジトリの入力を求められます。P Pとタイプするとリモートに現在のブランチをプッシュします。 言い換えれば、git push <remote> <branch>を実行します。そのブランチがリモートに存在しない場合は、リモートにそのブランチ作成します。ローカルブランチでは、その 新しく作ったリモートブランチをpullするよう設定します。P Pへの二重前置引数を与えると、プッシュ先のブランチの入力を求めます。言い換えれば、それはgit push <remote> <branch>:<target>を実行します 。
f fとタイプするとgit fetchを実行します。デフォルトのリモートが存在しない場合は、更新元のリモートの名前の入力を求めます。f oとタイプすると常にリモートの入力を求められます。 F Fとタイプするとgit pull を実行します。指定のリモートからpullする既定のブランチが設定されてないときは、それを尋ねられます。
現在のブランチの既定のリモートリポジトリがある場合は、Magitはステータスバッファのヘッダーにそのリポジトリを表示します。
この時、ステータスバッファには、ブランチ名が<remote>/<branch>となって無い現在位置のコミットを表示する、プッシュされてないコミットセ クションもあります。このセクションは履歴バッファのように動作します。RETを使ってコミットに関する詳細を見ることができ、.と=でそれらの2つを比較し、例えばxで、そ れらのいずれかに現在現在位置をリセットすることができます。 あなたが変更をプッシュする場合はP Pとタイプします。
リモートブランチに現在のブランチに含まれていない変更がある場合には、Magitは*Unpulled changes*と呼ばれるセクションでそれらを示しています。F Fとタイプするとフェッチして、それを現在のブランチにマージします。

---

21 サブモジュール

M uサブモジュールを更新しします。前置引数があると初期化になります。

M iサブモジュールを初期化します。

M bサブモジュール群の更新と初期化を一気に行ないます。

M s.gitmodulesで指定された値にサブモジュールのリモートURLの構成設定を同期します。

---

22 2分割表示

Magitはステータスバッファで、テスト前の段階やたくさんのリビジョンを表示するために分割表示をサポートします。分割表示セッションはステータスとログバッファからBキーによるメニューで制御することができます。
B sとタイプすると分割表示セッションを開始します。悪いことが判っているリビジョン(既定では*HEAD*)と、良いことが判っているリビジョン(デフォルトでは、もしあればポイントしているリビジョン)を提示します。gitがテストするためにリビジョンを選択すると、Magitはそれに応じてステータスバッファを更新します。
git に対して現在のリビジョンが良好な場合はB gとタイプし、悪い場合は{B bとタイプして教える事ができます。そのリビジョンをスキップさせるべきならB kです。テストの為に各リビジョンで走るスクリプトの名前を与えることによって全自動モード走るよう指示するにはB uとタイプします。
現在の状態はB lを使用して、ログとして表示することができます。それは既にテストされたリビジョンとあなたが設定した状態を含んでいます。
まだテストしていないリビジョンはB vとタイプしてgitkで可視化することができます。
分割表示を終了し、セッションをリセットしなければならないならB rとタイプします。

---

23 Magit拡張

23.1拡張機能の有効化

Magitはgit-svnやtopgitやstgitとの対話を可能にする拡張機能を同梱しています。これらの拡張機能の詳細な使用方法については、次項以降を参照してください。
拡 張機能は、グローバルに、またはリポジトリ単位でアクティブにすることができます。からこれらの拡張はマイナーモードとして実装され、1は、例えば使用す ることができます切り替えるには、Mx magit-topgitモード topgit拡張子を、対応するセクションおよびコマンド（UN）が利用できるようにします。
（そして、すべてのリポジトリの場合）、自動的にそれを実現するために、1つは、することができます例えば使用します。

```
(add-hook 'magit-mode-hook 'turn-on-magit-topgit)
```

Magitまた、gitに基づいて、さまざまな拡張機能を設定することができ、リポジトリ構成。

```
(add-hook 'magit-mode-hook 'magit-load-config-extensions)
```

これはgitの設定変数を読み込み、アクティブになります関連する拡張機能。
たとえば、次のコマンドを実行した後、topgit 拡張子は、すべてのリポジトリにロードされながら、svn 1唯一の現在の1にロードされます。

```
$ git config --global --add magit.extension topgit
 $ git config --add magit.extension svn
```

注意--add拡張するごとに独自のを取得することを意味するフラグを、のラインconfigファイル。

23.2 Subversionへのインターフェース

nを入力するとrは実行git svn rebase 、タイピングN cは実行をgit svn dcommitと入力N Fと実行git svn fetch 。N sは（数値、Subversion）は改訂を求められますし、その後コミットに対応するGitのsha1を検索します。これはリモートのSubversion リポジトリのパスに制限されています。接頭辞の付いた銅（Cu のN sユーザーは、検索するブランチを求めるプロンプトが表示されますインチ

23.3 Topgitへのインターフェース

Topgit（http://repo.or.cz /r/topgit.git）はパッチキューマネージャである使用することが容易になりますできるだけ近い生のGitにあることを目指し、Magit持 つ。特に、それはの異なるセットを使用する必要はありません"コミット"するためのコマンド、"更新"、...操作。
magit-topgit.elは、主によって、Magitとの基本的な統合を提供します"項目"を参照して提供します。
Topgit 支店は "T /"接頭辞を使用して、正規の方法で作成することができます慣例により。だから、"T / foo"のブランチを作成すると、実際に入力されますコミットした後、複数のブランチを持つ "トピックス"セクション。topdepsと。topmsg。
また、我々は引く方法は、（参照押し引き ）などの分岐があるそれは様々な依存関係を更新する必要がありますので、若干異なるその枝の。このような場合を除いて、大部分は透明でなければならない紛争の。

23.4 StGitへのインターフェース

StGit（http://www.procode.org /stgit）が提供するPythonアプリケーションで、キルトに似た機能（すなわち、プッシュ/に/からパッチをポップGitの上にスタック）。これ らの操作はGitのコマンドを使用して実行されとパッチがマージ容易できるように、Gitコミットオブジェクトとして格納されている標準Gitを使って他 のリポジトリにStGITのパッチの機能。
magit-stgit.elは、主によって、Magitとの基本的な統合を提供しますそのパッチはレギュラーとして見ることができる "シリーズ"のセクションを提供"訪問"行動を通じてコミットします。
あなたは、"適用"アクションと直列に現在のパッチを変更することができますだけでなく、あなたは、"廃棄"アクションを使用してそれらを削除することができます。
さらに、magit-stgit-refreshとmagit-stgit-rebaseのコマンドを使用すると、それぞれのStGitを実行してみましょう操作。

23.5拡張機能の開発

MagitはGit-関連との連携を可能にするための汎用的なメカニズムを提供しますそのような外国のVCS、パッチシステム、などのシステム...
特に、それは、以下のことができます。

- 現在の状態に関する特定の情報を表示するには、セクションを定義リポジトリの、既存のセクションに相対的にそれらを置く。
  magit- define-inserter自動的に呼ばれる2つのフックを定義しています
  magit-before-insert-SECTION-hookと magit-after-insert-SECTION-hook生成し、配置することができます複数のセクション。
  次の例では、我々への組み込み "スタッシュ"セクションを使用私たち自身の "foo"のいずれかを配置。

```
(magit-define-inserter foo ()
  (magit-git-section 'foo
```

```
                "Foo:" 'foo-wash-function
                "foo" "arg1" "arg2"))
        (add-hook 'magit-after-insert-stashes-hook 'magit-insert-foo)
```

- それらのセクションで新しいタイプのオブジェクトを定義します。
  関数foo-wash-function後工程それぞれ上記で定義されたは "foo arg1をarg2にはgit"コマンドの出力の行、およびすることができるである特定の行に型を関連付ける。
  単純な実装は次のようになります。

```
(defun foo-wash-function ()
  (let ((foo (buffer-substring (line-beginning-position) (line-end-position))))
    (goto-char (line-beginning-position))
    (magit-with-section foo 'foo
      (magit-set-section-info foo)
      (forward-line))))
```

  この場合、コマンド出力の各行は、に変換されます型のオブジェクト'foo 。
- にそれらを正しくディスパッチする汎用コマンドの動作を変更する関連するシステムは、必要に応じて新たに定義されたタイプを利用すること。

```
(magit-add-action (item info "discard")
  ((foo)
   (do-something-meaningful-for-discarding-a-foo)))
```

  これは、それらのオブジェクトに適用さkの動作を変更します。
- の存在を反映するように、基本的なコマンドに異なるロジックを差し込み拡張子。
  magit- define-command自動的に定義されmagit-CMD-command-hookへの関数のリストを含めることができます実際のコアコードの 前に呼び出します。実行は、最初にフックした後で停止しそれがnil以外の値を返します。これは、拡張ロジックの余地がある。

```
(add-hook 'magit-create-branch-command-hook 'foo-create-branch)
```

  関数foo-create-branch毎回呼び出される試みはブランチを作成させ、例えば、に反応することができる一定の命名規則。
- 新しいコマンドおよび関連するメニューを定義します。
  この部分はメニューがかかることを除けば、拡張機能に実際に固有のものではありません"機能拡張"サブメニューに配置します。

そ れはMagitエクステンションの作者がの慣例に固執することが示唆されたエクステンションのマイナーモードを作る。これは、など、多くの利点を持ってい ますユーザーが拡張子を切り替えることがあり、それは簡単だということが事実指定されたリポジトリ用の拡張子の特定のセットを設定します。
出荷の拡張機能を開発する方法の例として役立つことができる新しい拡張子。
基 本的にはfoo拡張子は、提供すべきmagit-foo-mode マイナーモードと同様に、turn-on-magit-foo関数。メインマイナーモードのタスクは、その様々なフックを登録/登録解除することです拡張子が必要です。一方、登録 されているアクションは、することができます放って、それらが上でのみ実行することができるため、全体的に活性化されマイナモードがオフのときに起こるこ とはありません表示された項目、。
呼び出すことを忘れないでくださいmagit-refreshマイナーモードがトグルされると対話的なので、関連するセクションを表示するか非表示にすることが可能となる。

---

24 直接Gitを使う

Magit はあなたが必要とするすべてをしない時の状況については、使用して、生のGitのコマンドを実行できます。{0}.{/0}(訳注:ピリオド)これは、 Gitコマンドの入力を求め、実行し、ステータスバッファを更新します。 出力は

$.をタイプすることにより閲覧することができます。

---

25 カスタマイズ

以下の変数は、あなたの作業にMagitを適応させるために使用することができます。
magit-git-executableGitの実行ファイルの名前。

magit-git-standard-optionsGitを実行するときの標準のオプション

magit-repo-dirsGitリポジトリを格納するディレクトリ。
MagitがGitリポジトリのディレクトリを探し、magit-statusが選ぶ為に提示します。
magit-repo-dirs-depthGitのリポジトリを探すために(訳注:ディレクトリの)最大の深さ。
magit-repo-dirsで指定するディレクトリでGitリポジトリを探しているとき 、Magitgはこの値で指定した深さまで探索します。
magit- save-some-buffersnilでないとき、magit-statusを実行する前に変更されたバッファを保存します。tlに設定するとバッファをセーブするかどうか確認し、'dontaskに設定すると変更されたバッファ全てを確認無しに保存します。

magit-save-some-buffers-predicate上の述語関数を指定magit-save-some-buffers保存されていないバッファを保存するためのプロンプトが表示されるべきかを決定するために。

magit- commit-all-when-nothing-staged何もステージされていない時magit-log-editga 内容を決定何も上演されていない場合に行います。に設定するとnilそれはそれを設定する場合は、何もしないでしょうt実際のcommitコマンドを使用 するように、物事手配いたし--allにそれを設定して、オプションを'ask最初にこれを行うかどうか確認を求めますし、それを設定する'ask- stage 、すべての変更が確認された後、上演されることになります。

magit-commit-signoffgit commitを実行するときに--signoffを付加します 。

magit-log-cutoff-lengthlogとwhazzupバッファで表示するコミットの最大数

magit-log-infinite-lengthの最大値として表示するために使用されるログの数magit-log-cutoff-length 。

magit-log-auto-more最後のエントリを通り過ぎた時は更に自動的にログエントリを追加します。
magit-goto-next-section の入力でリストの最後に達したときのみ働きます。
magit-process-popup-timeコマンド実行で、ここで指定した秒数より長い時間が必要な場合、プロセスバッファをポップアップします。

magit-revert-item-confirm項目を戻す前に承認を必要とします。

magit-log-edit-confirm-cancellationログの編集バッファを取り消す前に承認を必要とします。

magit-remote-ref-format何が形式リモコン用pariticularで、refsをオートコンプリートに使用する。
オートコンプリート機能はmagit-checkoutやmagit-interactive-rebaseや他のブランチ名補完のような機能で使われます。
'name- then-remoteの値はname (remote)形式でリモートリポジトリと解釈され、'remote-slash-nameの値はremote/name形式でリモートリポジトリと解 釈されます。すなわちリストにremotes/upstream/nextとあるものはgit branch -l -aによってupstream/next になります。
magit-process-connection-typegitのプロセスのために使用する接続タイプ。
nilはパイプを意味し、それは通常もっとも速く効率的で、かつcygwin上で動きます。tlはptyを意味し、必要な時にパスフレーズの入力を求めることをMagitlに許可します。
magit-completing-read-functionユーザーからの入力が必要な時に呼び出される関数。

magit-create-branch-behaviourブランチ名またはrefを指定しなかった場合、magitは新しいブランチを作成しま

す。
'at-headが意味するのは現在のブランチの先端で作成される新しいブランチです。'at-headの値によってmagitが有効な参照点が分かる限りは…
magit-status-buffer-switch-functionmagit-statusがステータスバッファの切り替えの為に使用する関数です。
関数は1つの引数を持ちます。引数にはステータスバッファが与えられます。
magit-rewrite-inclusivemagitが、rewrite操作内で選択されたベースコミットを含むかどうか。
値がtなら選択されたコミットと後続のコミットの両方が一緒に書き換えられることを意味します。これはmagitのデフォルトの振る舞いで、git rebase -i ${REV~1}に相当します。

```
A'---B'---C'---D'
 ^
```

値がnilなら 、選択したコミットが文字通りbaseとして使用されることを意味し、後続のコミット書き換えられます。これはステータスバッファから使われることがある更に面倒なgit rebase -i$(REV)に相当するgit-rebaseと等しいです。

```
A---B'---C'---D'
 ^
```

magit-topgit-executableTopGit実行可能ファイルの名前。

magit-topgit-branch-prefixトピックブランチを作成する時の共通の接頭辞。

---

26 よくある質問

26.1変更

- v1.1: 拡張子は仕事のやり方を変えました。ライブラリがロードされた一回以前に、彼らは無条件に有効になっていました。今、彼らは明示的に有効化する必要がマイナモード（潜在的にはリポジトリごとに）です。拡張機能の有効化 参照。

26.2トラブルシューティング

- FAQ 1-1 :どのように私はgitからの生のエラーメッセージを取得するのですか？

26.2.1質問1.1

どのように私はgitからの生のエラーメッセージを取得するのですか？

回答

コマンドがうまくいかなかった場合、$とタイプするとgitのプロセスバッファにアクセスすることができます。そこには、最後の操作のトレース結果が丸ごとあります。

26.3 表示問題

- FAQ 2-1 : 私は国際的な文字表示をどのように修正すればよいですか？

26.3.1質問2.1

私は国際的な文字表示をどのように修正すればよいですか？

回答

あなたのMagitバッファが互換性のあるコーディングシステムを使用していることを確認してください。ファイル名の特定のケースでは、それ自体をgitのデフォルトでは、それらを引用している。あなたは、次のいずれかの方法でこれを無効にすることができます。

```
$ git config core.quotepath false
```

または

```
(setq magit-git-standard-options (append magit-git-standard-options
                                         '("-c" "core.quotepath=false")))
```

後者は、gitの古いバージョンでは動作しない場合があります。