MIRACLE LINUX V4.0
セキュリティガイダンス

バージョン： 1.0

発行日： 2006 年 12 月 22 日

作成者： ミラクル・リナックス株式会社

目次

1. はじめに

本書は、MIRACLE LINUX V4.0 / MIRACLE LINUX V4.0 One / MIRACLE LINUX V4.0 x86-64 / MIRACLE LINUX V4.0 x86-64 One オペレーティングシステムのセキュリティに関するガイダンスである。

2. 使用にあたっての前提条件

2.1. 設置

本製品をインストールしたサーバー機器は、施錠されたサーバーラックやデータセンターなど、物理的な攻撃から保護される設備内に設置すること。

2.2. 管理者

本製品において、管理者は特別な権限を持ち、保存された情報の全てにアクセスすることが可能であるため、運用責任者は、悪意のある行為を行わず、与えられた権限に対する職務を果たす管理者を任命すること。

2.3. 利用者の教育

本製品へアクセスする際のパスワード、および自身が所有するオブジェクトのパーミッションは、該当利用者自身が変更することが可能であり、パスワードの品質や漏洩、所有するオブジェクトのパーミッションに対する責任は各利用者が負うこととなる。そのため、管理者は利用者に対し、パスワードやオブジェクトの管理についての教育を実施すること。

2.4. パスワード

TOE の利用者および管理者のパスワードは推測可能な文字列ではなく、かつ、各利用者および管理者は自身のパスワードを秘匿すること。

2.5. グループのパスワード

グループにパスワードを設定することにより、該当グループに属さない利用者であってもグループ切替えを実施することが可能となるが、セキュリティ上の問題からこの機能は利用しないこと。

2.6. グループの管理者

管理者がグループの管理者を設定することにより、グループの管理者はグループに所属する利用者を管理することが可能であるが、管理者はグループ管理者を設定する場合、信頼できる利用者を選定すること。

2.7. ログローテーションの設定

ログローテーションの設定は/etc/logrotate.conf にて行われているが、デフォルトでは期間のみ設定されている。ログのあふれ防止のため、管理者は容量によるログローテーションの設定を行うこと。

3. セキュリティ機能

3.1. 識別/認証

3.1.1. 識別/認証

TSF は、利用者および管理者が TOE へアクセスする前に必ず識別/認証を要求する。識別はユーザ ID に基づいて実施され、認証はユーザ ID に対するパスワード認証メカニズムによって実施される。また、パスワード漏洩対策のため、パスワード入力時は画面には入力に対するフィードバックは何も表示しない。

3.1.2. 識別/認証の管理

ユーザ ID の登録、削除、改変の操作は管理者のみに制限される。全ての利用者のパスワード登録は管理者のみに制限され、利用者は自身のパスワードのみを改変可能であり、管理者は自身および全ての利用者のパスワードを改変可能である。ただし、利用者が自身のパスワードを改変する際には、パスワード認証メカニズムにより再認証を要求する。パスワードは一定の品質を保つために一定の文字数以上を必要とし、その最小文字数の設定は管理者のみが変更可能である。

3.2. 任意アクセス制御

任意アクセス制御は、サブジェクトとオブジェクト間のアクセス制御を規定し、サブジェクトは常にプロセスであるが、オブジェクトは以下の 2 つが存在する。

- ファイルおよびディレクトリ
- プロセス

3.2.1. ファイルおよびディレクトリへの任意アクセス制御

オブジェクトがファイルおよびディレクトリの場合、ファイルおよびディレクトリは以下のセキュリティ属性を保持する。

＜ファイル＞

- ファイルの所有者
- ファイルの所有グループ
- ファイルの所有者のパーミッション
- ファイルの所有グループのパーミッション

- ファイルのその他の利用者のパーミッション
- ext2/ext3 ファイルシステムのファイルのアトリビュート（追加書き込み、変更不可）

＜ディレクトリ＞

- ディレクトリのスティッキービット
- ディレクトリの所有者
- ディレクトリの所有グループ
- ディレクトリの所有者のパーミッション
- ディレクトリの所有グループのパーミッション
- ディレクトリのその他の利用者のパーミッション
- ext2/ext3 ファイルシステムのディレクトリのアトリビュート（変更/削除不可、変更不可）

プロセスは、セキュリティ属性としてユーザ ID とグループ ID を保持する。これらセキュリティ属性と、上記のオブジェクトのセキュリティ属性に基づき、以下のアクセス制御規則を実施する。

- プロセスのユーザ ID が、該当ファイルまたはディレクトリの所有者と一致する場合、当該ファイルまたはディレクトリの所有者のパーミッションにて設定されている許可される操作が許可される。
- プロセスのグループ ID が該当ファイルまたはディレクトリの所有グループと一致する場合、当該ファイルまたはディレクトリの所有グループのパーミッションにて設定されている許可される操作が許可される。
- プロセスのユーザ ID、グループ ID に関わらず、該当ファイルまたはディレクトリのその他の利用者のパーミッションにて設定されている許可される操作が許可される。
- 該当ディレクトリに存在するファイルまたはディレクトリは、該当ディレクトリの所有グループのパーミッション、該当ディレクトリのその他の利用者のパーミッションによって、該当ディレクトリのディレクトリおよびファイルの生成/変更/削除が許可されていたとしても、スティッキービット属性が有効である場合は削除操作が拒否される。
- ext2/ext3 ファイルシステムのファイルのアトリビュート（追加書き込み）が有効である場合、該当ファイルのパーミッションによって該当ファイルの書き込み/追加書き込み操作が許可されている場合でも、追加書き込み操作だけに限定される。プロセスのユーザ ID が管理者であっても、追加書き込み操作だけに限定される。
- ext2/ext3 ファイルシステムのディレクトリのアトリビュート（変更/削除不可）が有効である場合、該当ディレクトリのパーミッションによって該当ディレクトリにおけるディレクトリまたはファイルの生成/変更/削除操作が許可されている場合でも生成操作だけに限定される。プロセスのユーザ ID が管理者であっても、生成操作だけに限定される。
- ext2/ext3 ファイルシステムのファイルのアトリビュート（変更不可）が有効である場合、該当ファイルのパーミッションによって該当ファイルの書き込み/追加書き込み操作が許可されている場合でも、書き込み/追加書き込み操作が拒否される。プロセスのユーザ ID が管理者であっても同

様に拒否される。

- ext2/ext3ファイルシステムのディレクトリのアトリビュート(変更不可)が有効である場合、該当ディレクトリのパーミッションによって該当ディレクトリの生成/変更/削除操作が許可されている場合でも、生成/変更/削除操作が拒否される。プロセスのユーザ ID が管理者であっても同様に拒否される。

ファイルの生成時においては、管理者が設定する以下のセキュリティ属性が付与される。

- ファイルの所有グループ
- ファイルの所有者のパーミッション
- ファイルの所有グループのパーミッション
- ファイルのその他の利用者のパーミッション

上記のセキュリティ属性はファイルを所有することになる所有者が、代替の初期値を設定することができる。

その他のセキュリティ属性は以下に示す値が設定される。

- 所有者: そのファイル、ディレクトリを生成したユーザ
- ext2/ext3 ファイルシステムのアトリビュート: 無効

ディレクトリの生成時において管理者が設定する以下のセキュリティ属性が付与される。

- ディレクトリの所有グループ
- ディレクトリの所有者のパーミッション
- ディレクトリの所有グループのパーミッション
- ディレクトリのその他の利用者のパーミッション

上記のセキュリティ属性はディレクトリを所有することになる所有者が、代替の初期値を設定することができる。

その他のセキュリティ属性は以下に示す値が設定される。

- 所有者: そのファイル、ディレクトリを生成したユーザ
- スティッキービット: 無効
- ext2/ext3 ファイルシステムのアトリビュート: 無効

ファイルにはデバイスファイル、UNIX ドメインソケット、名前付きパイプ、シンボリックリンクなど特殊なものが存在するが、これらは全て通常のファイルと同様の任意アクセス制御に基づいてアクセスが規定されるが、ext2/ext3 ファイルシステムのアトリビュートは保持しない。また、メモリ上のオブジェクトである IPC オブジェクト(共有メモリ、メッセージキュー、セマフォ)も、ファイルと同様の任意アクセス制御によりアクセスが規定されるが、実行パーミッションのセキュリティ属性は保持しない。

3.2.2. プロセスへの任意アクセス制御

全てのプロセスは、ユーザ ID を保持する。プロセスから他のプロセスおよびプロセスが保持するデータオブジェクトへのアクセス制御には、ユーザ ID だけが用いられ、サブジェクトのプロセスとオブジェクトのプロセスの関係は次のいずれかとなる。

- サブジェクトのプロセスのユーザ ID とオブジェクトのプロセスまたはプロセスが保持するデータオブジェクトのユーザ ID が一致する。
- サブジェクトのプロセスのユーザ ID とオブジェクトのプロセスまたはプロセスが保持するデータオブジェクトのユーザ ID が一致しない。

ユーザ ID が一致する場合、サブジェクトのプロセスはサブジェクト以外のプロセスに対して停止、復帰、削除、またはサブジェクト以外のプロセスが保持するデータオブジェクトへの読み込み、書き込みの操作を実行可能である。管理者はユーザ ID の値に依存せず、すべてのプロセスに対して上記操作が実行可能である。

3.2.3. セキュリティ属性の管理

サブジェクト、オブジェクトともに、プロセスのセキュリティ属性であるユーザ ID およびグループ ID は、そのプロセスを起動した利用者または管理者のユーザ ID、グループ ID が利用される。新しく生成するプロセスが、setuid ビット設定が有効であるファイルの実行に伴うものであれば、プロセスのユーザ ID は当該ファイルの所有者に設定変更される。また新しく生成するプロセスが、setgid ビットが有効であるファイルの実行に伴うものであれば、プロセスのグループ ID は当該ファイルの所有グループに設定変更される。

管理者は、プロセスのセキュリティ属性を改変可能である。利用者は、利用者を代行するプロセスのユーザ ID と一致するユーザ ID を持つプロセスに対して、グループ ID を利用者が所属するグループ ID へ変更可能である。setuid ビット設定、setgid ビット設定は、管理者及び所有者が改変可能である。

ファイルおよびディレクトリのセキュリティ属性は、所有者である場合、所有グループ、スティッキービットを含む各パーミッションを改変可能であり、管理者は所有者、所有グループ、スティッキービットを含む各パーミッション、ext2/ext3 ファイルシステムのアトリビュートを改変可能である。また、管理者は利用者によってファイルおよびディレクトリが生成される際に付与される所有グループ、所有者のパーミッション、所有グループのパーミッション、その他の利用者のパーミッションのデフォルト値を設定可能であり、所有者となる利用者は、これらを上書きする代替の初期値を設定可能である。管理者はファイルおよびディレクトリの所有者を改変可能である。

グループ ID の登録、削除、改変の操作は管理者のみに制限される。グループに所属する利用者の改変、削除、登録は管理者および該当グループ管理者に制限され、グループ管理者の改変、削除、登録は管理者のみに制限される。

3.3. 監査

3.3.1. 監査記録の生成

TOE は制御下で発生する監査事象の監査記録を生成する。監査事象は以下の通りである。

- 監査の起動と終了
- 利用者のログインにあたっての認証の成功および失敗
- 利用者のログイン中の利用者切り替えにあたっての認証の成功および失敗
- 管理者のログインにあたっての認証の成功および失敗
- 利用者のパスワード変更にあたっての再認証の失敗
- 利用者のログインにあたっての識別の成功および失敗
- 管理者のログインにあたっての識別の成功および失敗

TOE は、すべての監査事象に対して事象の日付/時刻、事象の種別、サブジェクト識別情報、事象の結果(成功または失敗)、その原因となった利用者の識別情報を記録することができる。監査記録に使用される日付/時刻は、TOE が提供する。

3.3.2. 監査記録の閲覧

監査記録は、管理者が閲覧可能な形式で提供され、管理者のみが閲覧可能である。TOE は監査分析のために事象の日付/時刻、事象の種別、サブジェクト識別情報、事象の結果(成功または失敗)に基づいた検索、分類、並び替えのツールを提供する。

3.3.3. 監査記録の管理

監査記録のファイルは管理者のみが読み込み、書き込み可能なパーミッションで生成され、不正な改変から防止している。記録する監査事象の粒度や事象種別、生成するファイル名の設定は管理者のみが変更可能である。管理者は監査記録消失の恐れに対しファイルの最大容量を設定可能であり、この設定を超えた場合には最も古い監査記録を削除し、新しい監査記録を生成する。監査機能は管理者のみが停止可能である。日付/時刻は、管理者のみが変更可能である。

4. セキュリティ機能のインターフェース

4.1. 識別/認証

4.1.1. ログイン時のインターフェース

管理者および利用者が TOE へアクセスする際には必ず識別/認証を実施する必要があるが、その手順は以下の通りである。

- 画面に「login: 」の文字列が表示される
- アカウント名を入力
- 画面に「password: 」の文字列が表示される
- パスワードを入力（入力されたパスワードは画面に表示されない）

管理者および利用者が、利用者の切替えを実施する際、管理者の場合は認証は行われず、利用者の場合はパスワードの入力が必要である。

4.1.2. パスワードの最小文字数

パスワードの最小文字数は管理者のみが変更可能であり、以下のファイルを編集することにより変更することができる。

- /etc/pam.d/system-auth

上記ファイル中の以下の行を

password requisite /lib/security/$ISA/pam_cracklib.so retry=3

次のように変更することで、最小文字数が 10-2 = 8 文字となる。

password requisite /lib/security/$ISA/pam_cracklib.so retry=3 minlen=10

pam_cracklib モジュールの minlen オプションは、設定した値-2 が実効値となることに注意する必要がある。

minlen 値を設定しない場合(デフォルトの状態)、パスワードの最小文字数は 6 文字に設定される。管理者はセキュリティ上、最小文字数を 5 文字以下に設定してはならない。

パスワードの最小文字数の制限は利用者がパスワードを変更する場合のみ有効であり、管理者の場合、警告は表示されるが制限は受けない。

4.2. ファイルおよびディレクトリへの任意アクセス制御

4.2.1. 所有者、所有グループ、パーミッション

ファイルおよびディレクトリは所有者、所有グループ、パーミッションを保持し、ls –l のコマンドにより確認することが可能である。

```
$ ls –l /etc/passwd
-rw-r--r--  1 root root 2325 10 月  12 10:23 /etc/passwd
```

1 番目のフィールドがパーミッション、3 番目、4 番目のフィールドがそれぞれ所有者、所有グループを示す。

パーミッションは 10 個の文字によって表される。一番左の 1 文字がファイルタイプであり、残りの 9 文字が 3 文字ずつのカテゴリ(所有者のパーミッション、所有グループのパーミッション、その他のパーミッション)を表す。各文字の意味を次に示す

ファイルタイプ	-	通常のファイル
	d	ディレクトリ
	l	シンボリックリンク
	b	ブロックデバイスファイル
	c	キャラクタデバイスファイル
	s	UNIX ドメインソケット
	p	名前付きパイプ
ファイルの場合の ・ 所有者のパーミッション ・ 所有グループのパーミッション ・ その他のパーミッション	r	ファイルの内容を参照可能
	w	ファイルに変更を加えることが可能
	x	ファイルを実行することが可能
	-	権限なし
ディレクトリの場合の ・ 所有者のパーミッション ・ 所有グループのパーミッション ・ その他のパーミッション	R	ディレクトリ内のファイル参照が可能
	W	ディレクトリ内にファイル作成が可能
	X	ディレクトリ内への移動が可能
	-	権限なし

上記の/etc/passwd ファイルの場合、次のようなセキュリティ属性となる。

- root ユーザが読み込み、書き込み可能である
- root グループに所属するユーザが読み込み可能である

- root ユーザではなく、root グループに所属しないユーザが読み込み可能である

また、このパーミッションはそれぞれ 3 桁の数値の表されることもあり、r=4、w=2、x=1、-=0 として合計した値で表す。上記の/etc/passwd ファイルの場合、所有者のパーミッション= rw- = 4+2+0 = 6、所有グループのパーミッション = r-- = 4+0+0 = 4、その他のパーミッション = r-- = 4+0+0 =4 として、644 と表される。

4.2.2. setuid/setgid

ファイルの特殊なパーミッションとして、setuid/setgid が存在し、ls -l コマンドで確認することが可能である。

```
$ ls -l /usr/bin/passwd
-r-s--x--x   1 root root 19796  8 月   3   2005 /usr/bin/passwd
```

所有者のパーミッションの実行権限に付与されている"s"が setuid を表し、setuid が不可されたプログラムを実行した場合、生成されるプロセスが保持するユーザ ID は、そのファイルの所有者と同一になる。上記の passwd コマンドの場合、どの利用者が実行を場合でも、生成されるプロセスの権限は root となる。

setuid と同様に、setgid の場合は生成されるプロセスのプロセスが保持するグループ ID が、そのファイルの所有グループと同一になる。

setuid/setgid パーミッションはプログラムに大きな権限を与えることも可能であり、使用方法を間違えるとセキュリティホールになることがあるため、使用に際しては注意すること。特に、所有者が root で setuid パーミッションが付加された場合、管理者権限でプログラムを実行できることから、必要がない限りそのようなパーミッション設定はすべきではない。

4.2.3. スティッキービット

ディレクトリの特殊なパーミッションとして、スティッキービットが存在し、ls -l コマンドで確認することが可能である。

```
$ ls -l /
drwxrwxrwt    11 root root   4096 11 月   9 22:42 tmp
```

その他のパーミッションの実行権限に付与されている"t"がスティッキービットを表し、スティッキービットが付与されたディレクトリ下に作成されたファイルは、書き込み権限のあるディレクトリ下のファイルであっても、所有者でない限り削除を行うことができない。

4.2.4. アトリビュート属性

ext2/ext3 上のファイルおよびディレクトリは、所有者、所有グループ、パーミッション以外にも専用の属性を保持し、lsattr コマンドで確認することが可能である。

```
$ lsattr
suS-iadAcj--- /root/test
```

上記の 1 番目のフィールドがファイルの属性であり、各文字の意味を以下に示す。

A	atime を更新しない
S	同期更新
D	ディレクトリの同期更新
a	追加のみ許可
c	圧縮
d	ダンプしない
i	変更不可
j	データのジャーナリング
s	安全な削除
u	復活可能

4.2.5. ファイルおよびディレクトリ新規作成時のデフォルトのパーミッション

bash においてファイルおよびディレクトリを作成した際のデフォルトのパーミッションは、それぞれ 666、777 となる。しかしながら、umask を設定することでデフォルト値を設定することができ、管理者は /etc/bashrc に umask 値を設定することで、全ての利用者、管理者のデフォルトパーミッションを制御することが可能である。umask 値が設定されている場合、作成時のパーミッションはデフォルト値から umask 値を引いた値となる。

TOE はデフォルトで/etc/bashrc に umask 022 の設定がされており、ファイルおよびディレクトリのデフォルトのパーミッションは、それぞれ 644、755 となる。

この管理者が設定した値は、利用者が任意に変更可能であり、ホームディレクトリの.bashrc または.bash_profile に umask 設定値を記述するか、またはログイン後にコマンドプロンプトから直接 umask コマンドを実行することにより上書き設定が可能である。

ただし、ファイルに対して実行権限を付与する umask 値を設定した場合、該当パーミッションは+1 された値が利用され、安易に実行権限の付与されたファイルが作成されないようになっている。

4.2.6. 特殊なファイルのセキュリティ属性

特殊なファイルについて、設定可能なセキュリティ属性の一覧を示す。

		デバイスファイル	UNIX ドメインソケット	名前付きパイプ	シンボリックリンク（注 2）	IPC オブジェクト
所有者		○	○	○	----	○
所有グループ		○	○	○	----	×
パーミッション	読込	○	○	○	----	○
	書込	○	○	○	----	○
	実行	○	○	○	----	×
setuid/setgid		○	○	○	----	×
アトリビュート		×	×	×	----	×
スティッキービット（注 1）		----	----	----	----	----

注 1: 各ファイルにはディレクトリは存在しないためスティッキービットは設定不可

注 2: シンボリックリンクのセキュリティ属性は実体のファイルに従う

4.2.7. インターフェース

ファイルおよびディレクトリへの任意アクセス制御は、カーネルによって実装されているため、管理者や利用者がアクセス制御機能自体の操作を行うためのインターフェースは存在しない。管理者や利用者がファイルおよびディレクトリを操作するとき、その命令は最終的にカーネルが受け取り処理を行うため、結果として必ずアクセス制御が実施される。

4.3. プロセスへの任意アクセス制御

4.3.1. 所有者、所有グループ

プロセスはユーザ ID を保持し、ps コマンドによって確認することができる。

```
$ ps aux
root    4812    0.0    0.0    1652    616    ?    SNs    Nov12    0:00    /sbin/syslogd
```

1 番目のフィールドがユーザ ID を示す。ユーザ ID が一致した場合、もしくは管理者のみが、プロセスの停止、復帰、削除、読み込み、書き込みを行うことができる。

4.3.2. インターフェース

プロセスへの任意アクセス制御は、カーネルによって実装されているため、管理者や利用者がアクセス制御機能自体の操作を行うためのインターフェースは存在しない。管理者や利用者がファイルおよびディレクトリを操作するとき、その命令は最終的にカーネルが受け取り処理を行うため、結果として必ずアクセス制御が実施される。

4.4. 監査記録の生成

監査記録は syslog および klog によって記録される。syslog は識別/認証やその他の TOE 上で動作するアプリケーションの監査記録を保存する。klog はカーネルの監査記録を保存する。

監査記録機能の起動は、TOE 起動時に自動的に行われる。TOE 起動後の監査記録機能の起動、停止は以下のコマンドを使用することにより管理者のみが実行可能である。

- 開始: /etc/init.d/syslog start
- 停止: /etc/init.d/syslog stop

4.5. 監査記録の管理

監査記録のファイルは、以下の所有者、所有グループ、パーミッションが設定され、管理者以外の利用者からの不正なアクセスを防ぐ。

- ファイル名: /var/log/messages
- 所有者: root
- 所有グループ: root
- パーミッション: 600

監査記録のファイルは、cron デーモンが logrotate コマンドを実行することにより、/etc/logrotate.conf に設定された期間、容量、保存ファイル数を元にローテーションが行われる。デフォルトでは/etc/cron.daily ディレクトリに logrotate コマンドを実行するシェルスクリプト logrotate が置かれているため、1 日に 1 回の間隔で実行される。

デフォルトの logrotate.conf の設定では、作成から 1 週間以上経過した監査記録のファイルをリネームし、4 世代以上あれば最も古いファイルを削除し、新しいファイルを作成する。

4.6. ユーザー管理

4.6.1. ユーザーの追加

管理者がユーザーを新規追加する場合は、useradd コマンドでユーザーを作成し、passwd コマンドで初

期パスワードを設定する必要がある。

4.6.2. ユーザーの削除

管理者がユーザーを削除するとき、userdel コマンドを実行するだけではホームディレクトリが削除されないため、、完全に削除するためには-r オプションを付加するか、手動でホームディレクトリを削除する必要がある。

4.6.3. パスワードの再設定

パスワードを紛失した場合は、管理者のみが再設定を行うことができる。

5. 重要なファイルのセキュリティ属性

以下のファイルはセキュリティの観点から編集を管理者のみに制限すべきファイルであり、デフォルトの設定から変更してはならない。

ファイル名	デフォルトのセキュリティ属性		
	所有者	所有グループ	パーミッション
/etc/passwd	root	root	644
/etc/group	root	root	644
/etc/shadow	root	rppt	600
/etc/gshadow	root	root	600
/etc/pam.d/system-auth	root	root	644
/etc/sudoers	root	root	440
/etc/bashrc	root	root	644
/etc/syslog.conf	root	root	644
/etc/logrotate.conf	root	root	644
/home/<ユーザ ID>/.bashrc	ファイルの所有者	ファイルの所有者のプライマリグループ	644
/var/log/messages	root	root	600
/var/run/klogd.pid	root	root	600
/var/run/syslogd.pid	root	root	600
/etc/syslogd.conf	root	root	644

6. コマンド、システムコール、ライブラリコール

この章では、セキュリティに関わるコマンド、システムコール、ライブラリコールの利用方法を記載する。なお、コマンドの返り値は、成功した場合は 0、失敗した場合は 0 以外の値であり、それ以外の場合は各コマンドの説明中に詳細を示す。

6.1. 利用者コマンド

6.1.1. ログイン

6.1.1.1 システム上でセッションを開く (login)

書式

```
login [-p] [username] [ENV=VAR ...]
login [-p] [-h host] [-f username]
login [-p] -r host
```

説明

login はシステムに新たにセッションを開くために用いられる。通常は、ユーザの端末に表示される login: というプロンプトに応じる事によって自動的に起動される。login はシェル専用のものであり、サブプロセスとして起動することはできない。通常シェルは login を exec login とみなすので、ユーザは現在のシェルから抜けることになる。ログインシェル以外から login を起動しようとすると、エラーメッセージが表示される。

login:プロンプトから起動した際は、ユーザ名に続いて環境変数を入力する事もできる。それらを入力する場合は NAME=VALUE という書式で行う。この方法で全ての変数を設定できるわけではない。例えば PATH、HOME、SHELL などは設定できない。さらにログインシェルが/bin/sh の場合は IFS も設定もできない。

次いで、必要な場合には、ユーザはパスワードを入力するよう促される。パスワードを表示してしまわないよう、エコーは行われない。数回以上パスワード入力に失敗すると login は終了し、通信の接続は切断されてしまう。

アカウントに対してパスワードの有効期限が設定されている場合は、先に進む前に新しいパスワードの設定を促されることもある。セッションを続けるためには古いパスワードと新しいパスワードを入力しなくてはならない。詳しい情報は passwd(1)を参照すること。

ログインに成功すると、システムメッセージやメールの有無が表示される。ログインディレクトリに長さ0のファイル.hushlogin を作っておけば、システムメッセージファイルである/etc/motd の表示を無効にできる。メールに関するメッセージは、メールボックスの状態によって"You have new mail."、"You have mail."、"No Mail."のいずれかになる。

ユーザ ID とグループの ID は/etc/passwd ファイル中に記載されている値に従って設定される。$HOME、$SHELL、$PATH、$LOGNAME、$MAIL の値は、パスワードエントリのそれぞれのフィールドに従って設定される。ulimit, umask、nice 値が、GECOS フィールドのエントリーによって設定されることもある。

インストール時の設定によっては、/etc/ttytype の指定に従って、環境変数$TERM が tty 接続の端末の型(terminal type)に初期化されることもある。

コマンドインタプリタの初期化スクリプトが実行されることもある。この機能についての詳しい情報は適当なマニュアルセクションを参照のこと。

サブシステムログインでは、ログインシェルの最初の文字に"*"を置く。渡されたホームディレクトリは、ユーザが実際にログインする新しいファイルシステムのルートとして扱われる。

オプション

- -p
  環境を保存する。
- -f
  ユーザはすでに認証されているものとして、認証動作を行なわない。
- -h
  このログインのリモートホストの名前。
- -r
  rlogin の自動ログインプロトコルを実行する。

-r、-h、-f オプションは、root が login を起動した場合にのみ用いる。

警告

この版の login には多くのコンパイル時オプションがあるが、サイトによってはこのうちの一部しか使われていないかもしれない。

システム設定の違いによって上記ファイルの置き場所は変わる。

ファイル

- /etc/utmp - 現在のログインセッションのリスト

- /etc/wtmp – 過去のログインセッションのリスト
- /etc/passwd – ユーザアカウント情報
- /etc/shadow – 暗号化パスワードと有効期限情報

/etc/motd – システムメッセージファイル

- /etc/nologin – root 以外のユーザのログインを禁止する
- /etc/ttytype – 端末の型のリスト
- $HOME/.profile – デフォルトシェルの初期化スクリプト
- $HOME/.hushlogin – システムメッセージの表示を抑制する

関連項目

mail(1)、passwd(1)、sh(1)、su(1)、login.defs(5)、nologin(5)、passwd(5)、getty(8)

6.1.2. パスワード管理

6.1.2.1 ユーザパスワードを変更する (passwd)

書式

```
passwd [-f|-s] [name]
passwd [-g] [-r|-R] group
passwd [-x max] [-n min] [-w warn] [-i inact] login
passwd {-l|-u|-d|-S|-e} login
```

説明

passwd はユーザアカウント・グループアカウントのパスワードを変更する。一般ユーザは自分のアカウントのパスワードしか変更できない。スーパーユーザはいかなるアカウントのパスワードも変更できる。グループの管理者はグループのパスワードを変更できる。passwd によって、ユーザのフルネーム・ログインシェル・パスワードの期限切れの日付・有効期間といったアカウント情報を変更することもできる。

-s オプションを指定すると passwd は chsh を呼び出してユーザのシェルを変更する。-f オプションを指定すると passwd は chfn を呼び出してユーザの GECOS 情報を変更する。これらの 2 つのオプションは互換性のためだけにある。chsh や chfn を直接呼び出しても構わない。

パスワードの変更

パスワードが既にある場合は、まず古いパスワードを入力するよう促される。入力されたパスワードは暗号化され、記録されているものと照合される。正しいパスワードを 1 回で入力しなくてはならない。スーパーユーザは、パスワードを忘れてしまった際の変更も行なえる様に、このステップを省略できる。

パスワードが入力された後、パスワード有効期限の情報を調べ、現在パスワードの変更が許されているか検査する。もし許可されていない場合は、passwd は変更を拒否して終了する。

次にユーザは、置き換えるパスワードを入力するよう促される。入力されたパスワードは、充分複雑かどうか検査される。一般的な指針としては、パスワードは以下の集合それぞれから一つ以上の文字を使った 6 から 8 文字のものにすべきである。

- 小文字のアルファベット
- 大文字のアルファベット
- 0 から 9 までの数字
- 句読点

システムのデフォルトの消去文字や kill 文字を含めないように注意すること。passwd はあまりに単純なパスワードへの変更は拒否する。

入力したパスワードが受け入れられた場合、passwd はもう一度入力を促し、二番目に入力したものを最初のものと比較する。パスワード変更が受け入れられるためには、この両者が合致しなくてはならない。

グループパスワード

-g オプションを用いた場合、指定したグループのパスワードが変更される。このオプションはスーパーユーザか指定したグループの管理者しか使えない。現在のグループパスワードは尋ねてこない。-g オプションを-r オプションとともに用いると、指定したグループのパスワードが削除される。こうすると全てのメンバーがこのグループにアクセスできるようになる。-R オプションを-g オプションとともに用いると、全てのユーザに対して指定したグループへのアクセスを禁止できる。

パスワードの有効期限情報

スーパーユーザは、パスワードの有効期限に関する情報を変更できる。これには-x,-n,-w,-i などのオプションを用いる。-x オプションはパスワードが有効な最長日数を設定するのに用いられる。max 日が過ぎるとパスワードを変更するように求められる。-n オプションはパスワードが変更可能となるまでの最短日数を設定するのに用いられる。ユーザは min 日が経過した後でないとパスワードを変更できない。-w オプションはパスワードの使用期限が来る前に何日間警告を与えるかを設定するために用いられる。期限切れの warn 日前から注意が開始され、パスワードが期限切れになるまであと何日残っているかが示される。-i オプションは、パスワードの期限が切れてから何日間経過したら、そのアカウントを使用不能の状態にするかを設定するのに用いる。inact 日間アカウントをパスワード期限切れ状態のままにすると、ユーザはそのアカウントに入れなくなる。

あるアカウントのパスワードを直ちに期限切れにしたい場合は、-e オプションを用いればよい。するとそのユーザは次にログインする際にパスワードを変更するよう強制される。-d オプションを使って、ユーザのパスワードを削除することもできる(パスワードが空になる)。このオプションは注意して使うこと。これを

使うと、そのアカウントはログインにパスワードを全く必要としなくなり、システムが侵入者に対してオープンになってしまう。

アカウントの保守

-l フラグと-u フラグを用いると、ユーザアカウントをロックしたり、そのロックを外したりできる。-l オプションを用いると、パスワードフィールドの値は暗号化された如何なる値ともマッチしなくなり、アカウントは使用不能になる。-u オプションを用いると、パスワードは以前の値に戻り、アカウントが再び使用可能となる。

-S オプションを用いるとアカウントの状態が表示される。アカウントの状態の情報は 6 つの部分からなる。最初の部分は、アカウントにロックがかけられている(L)、パスワードが存在しない(NP)、もしくは使用可能なパスワードがある(P)といった情報を示す。2 番目は最後にパスワードが変更された日付を示す。残りの 4 つの部分はそれぞれパスワードの最短期限、最長期限、警告期間、使用不能期間である。

ユーザパスワードに対するヒント

パスワードの安全性は暗号化アルゴリズムの強力さとキー空間の大きさに依存する。UNIX のシステム暗号化の方法は NBS DES アルゴリズムに基づいており、非常に安全性が高い。キー空間の大きさは選ばれたパスワードのランダムさに依存する。

パスワードの安全性が脅かされるのは、大抵の場合パスワードの選択や扱いが不注意なためである。従ってパスワードとしては、辞書に載っているものや書き留めなければならないものは避けるべきである。また、固有名詞・免許証番号・誕生日・自宅の住所などをパスワードにするのも避けるべきである。これらはいずれもシステムセキュリティを破る際に、推量情報に用いられる可能性があるからである。

パスワードは紙片に書き留めておく必要が無いよう、簡単に思い出せるものにしなくてはならない。これは例えば、短い二つの単語をくっつけて、その間に特殊記号や数字を挟み込むことによって作れる。例えば Pass%word など。

他の作り方としては、文学作品などから思い出しやすい句を選び出し、それぞれの単語から最初もしくは最後の文字を抜き出す方法がある。この方法の例としては、Ask not for whom the bell tolls. という句から An4wtbt.というパスワードが作り出せる。

クラッカーの辞書には、こんな語句は載っていなさそうだ、とみなしても良いだろう。しかし、ここに示した方法だけに頼るのではなく、自分独自のパスワードの作り方を考え出すべきである。

グループのパスワードに関する注意

グループパスワードは、一人以上の人間が知ることが許されるものであるから、本質的にセキュリティ上の問題を抱えている。しかしグループを使えば別々の人間が共同で作業する事ができるので、これは便利なツールではある。

警告

全てのオプションが使えるようには設定されていないかもしれない。パスワードの複雑さの検証はサイトによって異なるだろう。ユーザはシステムが満足するような、充分複雑なパスワードを選ぶよう強制される。NIS が動作していて、かつ NIS サーバ以外にログインしているユーザは、パスワードを変更できない。(訳注:この場合 yppasswd(8)を用いる。)

ファイル

- /etc/passwd – ユーザアカウント情報
- /etc/shadow – 暗号化されたユーザパスワード

関連項目

group(5)、passwd(5)

6.1.3. 利用者切替え

6.1.3.1 ユーザ ID とグループ ID を変更してシェルを起動する (su)

書式

su [-flmp] [-c command] [-s shell] [--login] [--fast] [--preserveenvironment] [--command=command] [--shell=shell] [-] [--help] [--version] [user [arg...]]

説明

この文書はもうメンテナンスされていないので、不正確・不完全な可能性がある。現在は texinfo 文書が正式な情報となっている。

このマニュアルページは GNU 版 su について記述したものである。su はあるユーザーが一時的に他のユーザになるために用いられる。su は user の実ユーザーID、実効ユーザーID、グループ ID、および所属グループの権限を与えてシェルを起動する。もし user が与えられなかった場合、デフォルトは root、すなわちスーパーユーザーである。実行されるシェルは user のパスワードエントリから選択される。ここに何も書かれていない場合は/bin/sh が実行される。user にパスワードがある場合には、su はパスワードを要求するプロンプトを表示する。ただし su を実行したのが実ユーザーID 0(スーパーユーザー)の場合にはパスワード認証は行われない。

su はデフォルトではカレントディレクトリを変更しない。また su は環境変数 'HOME' および 'SHELL' を user のパスワードエントリの値にセットする。また user がスーパーユーザー以外の場合には、環境変数

‘USER’と ‘LOGNAME’を user にセットする。デフォルトでは、起動されるシェルはログインシェルにはならない。

ひとつ以上の引き数 arg が与えられた場合には、これらはシェルに渡され、引き数としてシェルに付加される。

su は/bin/sh あるいはそれ以外のいかなるシェルも特別には扱わない（argv[0]に"-su"を設定したり、あるシェルにだけ-c を渡したり、といったことはしない）。

syslog に対応しているシステムでは、su が失敗したとき syslog にレポートするようにコンパイルすることができる（成功をレポートするようにもできる）。

このプログラムは"wheel group"の機能（su によってスーパーユーザーアカウントになれるユーザを制限する機能）をサポートしない。これは専制的なシステム管理者が他のユーザーに不当な権力を振るえないようにするためである。

オプション

- -c COMMAND、--command=COMMAND

  対話的なシェルを起動するのではなく、シェルに-c オプションとともに COMMAND（実行されるコマンドライン一行）を渡す。
- -f、--fast

  シェルに-f オプションを渡す。これは（おそらく）csh と tcsh のみで意味を持つ。これらのシェルでは-f オプションを指定すると、スタートアップファイル（.cshrc）を読み込まない。Bourne 系のシェルでは-f オプションはファイル名パターンの展開を抑制する。これは大抵の場合は望ましい動作ではないだろう。
- --help

  使い方に関するメッセージを標準出力に表示し、実行成功を返して終了する。
- -、-l、--login

  シェルをログインシェルにする。すなわち以下のような取り扱いをする:すべての環境変数を解除する。その上で‘TERM’、‘HOME’、‘SHELL’を前述のように設定し、‘USER’、‘LOGNAME’（スーパーユーザーであっても）を同じく前述のように設定する。続いて‘PATH’をコンパイル時のデフォルト値に設定する。ディレクトリを user のホームディレクトリに変更する。シェル名の前に‘-’を付加し、シェルにログイン時のスタートアップファイルを読ませる。
- -m、-p、--preserve-environment

  ‘HOME’、‘USER’、‘LOGNAME’および‘SHELL’環境変数を変更しない。/etc/passwd で指定されている user のシェルではなく、現在の環境変数‘SHELL’で指定されているシェルを実行する。ただし su を実行するユーザーがスーパーユーザではなく、user によるシェルの実行が制限されている場合はこの限りではない。実行が制限されているシェルとは、/etc/shells に

リストされていないシェル、あるいは/etc/shells が無い場合はコンパイル時の指定リストに存在しないシェルのことである。このオプションが実行する作業の一部は--login または--shell によってオーバーライドされる。

- -s、--shell shell

  /etc/passwd に記述された user のシェルの代わりに shell を実行する。

- --version

  バージョン情報を標準出力に表示し、実行成功を返して終了する。

GNU su で wheel グループをサポートしないわけ（Richard Stallman）

ときおり、少数のユーザーによって、他のユーザーに対する全権を掌握しようとする試みがなされることがある。例えば 1984 年、MIT AI ラボの少数のユーザーは Twenex システムのオペレーターパスワードの変更権限を強奪し、これを他のユーザーから秘匿することに決定した（この際には私はこのクーデターの裏をかき、カーネルにパッチを当てて権限を取り返すことに成功した。しかしこれが Unix であったら、私にはどうすればよいかわからなかっただろう）。

しかしながら、時には専制者も秘密を漏らすものである。通常の su のメカニズムでは、一般ユーザーの側に立つ者が root のパスワードを知れば、これを他のユーザーにも知らせることができる。しかし"wheel group"機能はこれを不可能にし、結果として専制者達の権限を強固たるものにしてしまう。

私は大衆の側に立つものであり、専制的な立場には反対する。あなたはボスやシステム管理者のやり口に従うことに慣れているかも知れないが、その場合はまずそのこと自身を不思議に思うべきではないだろうか。

6.1.3.2 別のユーザとしてコマンドを実行する（sudo）

書式

sudo -V | -h | -l | -L | -v | -k | -K | -s | [-H] [-P] [-S] [-b] | [-p prompt] [-c class | -] [-a auth_type] [-u username | #uid] command

説明

sudo は、許可されたユーザに対して、スーパーユーザや別のユーザの権限で command を実行することを許す。この指定は sudoers ファイルでなされる。実ユーザ ID・グループ ID と実効ユーザ ID・グループ ID は、成り代わるユーザのものと置き換えられる。passwd ファイルでの指定が用いられる。（成り代わるユーザが root でない場合、ユーザの入っているグループも初期化される）。デフォルトでは、sudo はパスワードを使った自分自身に対する認証が必要とする(注意：このときのパスワードはそのユーザのパスワードであり、root パスワードではない)。一度ユーザが認証されると、タイムスタンプが更新され、短い期間

(sudoers で上書きされない限り、デフォルトでは 5 分間)パスワードなしで sudo を使うことができる。

sudo は、権限のあるユーザが誰かを/etc/sudoers ファイルによって決定する。sudo に-v フラグをつけて実行すると、command を実行することなく、タイムスタンプを更新できる。パスワードプロンプト自身も、ユーザのパスワードが 5 分間入力されないとタイムアウトする(sudoers で上書きされない限り)。

sudoers ファイルに記載されていないユーザが sudo を使ってコマンドを実行すると、誰か偉い人にメールが送られる。偉い人を誰にするかは(コンパイルの)設定時または sudoers ファイルで定義される(デフォルトでは root)。権限のないユーザが-l または-v フラグをつけて sudo を実行した場合は、メールが送られないことに注意すること。これにより、ユーザは自分が sudo が使用可能であるかを自分自身で調べることができる。

sudo は成功した命令・失敗した命令の両方を(あるいはエラーも)、syslog(3)やログファイル(あるいはその両方)に記録できる。デフォルトでは sudo は syslog(3)を使ってログをとる。これはコンパイルの設定時または sudoers ファイルで変更できる。

オプション

sudo は、以下のコマンドラインオプションを受け付ける

- -V -V (version)
  オプションが指定されると、sudo はバージョン番号を表示して終了する。このコマンドを呼び出したユーザが既に root であった場合、-V では sudo をコンパイルした時のデフォルト値のリストと、マシンのローカルネットワークアドレスを表示する。
- -l -l (list)
  オプションが指定されると、そのユーザ対して現在のホスト上で許可された(禁止された)コマンドがリストされる。
- -L -L (list defaults)
  オプションが指定されると、Defaults 行に設定できるパラメータが短い説明をつけてリストされる。このオプションは、grep(1)と一緒に使うと便利である。
- -h -h (help)
  オプションが指定されると、sudo は使用法のメッセージを表示して終了する。
- -v -v (validate)
  オプションが指定されると、sudo はユーザのタイムスタンプを更新する。必要ならば、ユーザのパスワードを問い合わせるプロンプトを出す。このオプションは、コマンドを実行することなく、(sudoers でタイムアウトが何分に設定されていても)sudo のタイムアウトを更に 5 分間延長する。
- -k -k (kill)
  オプションが指定されると、sudo は有効期間を紀元年 (epoch)に設定することで、ユーザのタ

イムスタンプを無効にする。次回 sudo を実行するときは、パスワードが必要とされる。このオプションにはパスワードが必要ない。ユーザが.logout ファイルで sudo 権限を取り消すことができるように追加された。

- -K -K (sure kill)

  オプションが指定されると、sudo はユーザのタイムスタンプを完全に削除する。このオプションにはパスワードが必要ない。

- -b -b (background)

  オプションが指定されると、sudo は指定されたコマンドをバックグラウンドで実行する。-b を使った場合、プロセスの操作にシェルのジョブ制御を使うことができない点に注意すること。

- -p -p (prompt)

  オプションが指定されると、デフォルトのパスワードプロンプトを上書きして、カスタム化したものを使うことが可能になる。プロンプトに%u エスケープがある場合、%u はユーザのログイン名に置き換えられる。同様に、%h はローカルホスト名に置き換えられる。

- -c -c (class)

  オプションが指定されると、sudo は指定されたコマンドをログインクラスで指定されたリソースの制限内で実行する。引き数 class には、/etc/login.conf で定義されているクラス名を指定するか、あるいは 1 個の'-' 文字を指定することができる。class を-に指定すると、コマンドは、そのコマンドを実行されたユーザのデフォルトのログイン権限によって制限を受ける。引き数 class が存在しているユーザクラスを指定している場合、コマンドは root として実行されなければならない (もしくは sudo コマンドを既に root になっているシェルから実行しなければならない)。このオプションは、BSD ログインクラスのあるシステムでのみ有効で、かつ sudo に--with-logincap オプションが設定されていることが必要である。

- -a -a (authentication type)

  オプションが指定されると、sudo はユーザの認証に/etc/login.conf で許可されている認証タイプを使用する。システム管理者は/etc/login.conf に"auth-sudo"エントリを追加することにより、sudo 独自の認証法を指定することができる。このオプションは BSD 認証をサポートするシステムで、sudo に--with-bsdauth オプションが指定されて(コンパイルされて)いる場合にしか使用できない。

- -u -u (user)

  オプションが指定されると、sudo は root 以外のユーザとして指定したコマンドを実行する。username でなく uid を指定する場合は、#uid を使うこと。

- -s -s (shell)

  オプションが指定されると、環境変数 SHELL が設定されている場合は、そのシェルを実行する。さもなければ、passwd(5)で指定されているシェルを実行する。

- -H -H (HOME)

  オプションが指定されると、環境変数 HOME が passwd(5)で指定された対象ユーザ(デフォルト

では root)のホームディレクトリに設定される。デフォルトでは、sudo は HOME を変更しない。

- -P -P (preserve group vector)

  オプションが指定されると、sudo はユーザのグループリストを変更しない。デフォルトでは、sudo はグループリストを対象ユーザが所属するグループのリストで初期化する。ただし実グループ ID と実効グループ ID は、対象ユーザにマッチするように設定される。

- -S -S (stdin)

  オプションが指定されると、sudo は端末デバイスではなく標準入力からパスワードを読む。

- --

  -- オプションは、sudo がコマンドライン引き数の処理を終了することを示している。-sオプションと一緒に使うと、とても便利である。

返り値

プログラムが正常に実行されると、sudo の返り値は単純に実行されたプログラムの返り値になる。

設定やアクセス権の問題があった場合、もしくは、sudo が指定されたコマンドを実行できなかった場合、sudo は終了値 1 で終了する。後者の場合は、エラー文字列が標準エラーに表示される。sudo がユーザの PATH のエントリのどれかを stat(2)できない場合、エラーが標準エラーに表示される(ディレクトリが存在しない場合、またはエントリが実際のディレクトリでない場合は、エントリは無視され、エラーは表示されない)。これは通常の環境では起こらない。stat(2)が"permission denied"を返す最も一般的な理由は、automounter を稼働させているときに、PATH にあるディレクトリのどれかが現在アクセスできない計算機上にあるというケースである。

セキュリティ上の注意

sudo は外部コマンドを実行する場合、安全であるように努める。動的ロードや動的バインドを制御する環境変数を使って、sudo が実行するプログラムの安全性を下げようとすることができる。これに対抗するため、環境変数 LD_*,_RLD_*、SHLIB_PATH(HP-UX のみ)、LIBPATH(AIX のみ)は、実行されるコマンドに渡される環境変数からは、いかなる場合も削除される。¥fBsudo¥fR は、環境変数 IFS、ENV、BASH_ENV、KRB_CONF 、 KRBCONFDIR 、 KRBTKFILE 、 KRB5_CONFIG 、 LOCALDOMAIN 、 RES_OPTIONS 、 HOSTALIASES、NLSPATH、PATH_LOCALE、TERMINFO、TERMINFO_DIRS、TERMPATH も同様な脅威を引き起こすので削除する。TERMCAP にパス名が設定されている場合も無視される。さらに LC_*、LANGUAGE 変数に/、%文字が含まれている場合も無視される。sudo が SecurID をサポートするようにコンパイルされている場合、VAR_ACE、USR_ACE、DLC_ACE 変数も削除される。sudo が削除する環境変数のリストは、root で sudo -V を実行したときに表示される。

コマンドスプーフィング(だましコマンド)を阻止するため、ユーザの PATH でコマンドを検索するときに、sudo は(カレントディレクトリを意味する)"."と""(のいずれかもしくは両方が PATH にあるかどうか)を最後

にチェックする。しかし、実際の環境変数 PATH は修正されず、そのまま sudo が実行するプログラムに渡されることに注意すること。

使用している OS が共有ライブラリをサポートしているのに、setuid プログラムに対してユーザ定義のライブラリ検索パスを許している場合(ほとんどはそうである)、セキュリティのため、これを許さないようにするリンカオプションを使うか、sudo を静的にリンクすべきである。

sudo はタイムスタンプディレクトリ(デフォルトでは/var/run/sudo)の所有者をチェックし、所有者が root で、かつ root のみの書き込み属性でない場合、ディレクトリの中身を無視する。root 以外のユーザでも chown(2)を使って自分のファイルを他人に渡せるようなシステムでは、タイムスタンプディレクトリが全てのユーザに書き込み可能なディレクトリ(例えば、/tmp)である場合、ユーザが sudo の実行前にタイムスタンプディレクトリを作成できてしまう。しかし sudo はディレクトリと中身の所有者とアクセス権をチェックするので、受けるダメージとしては「隠し」ファイルをタイムスタンプディレクトリに入れられるだけである。タイムスタンプディレクトリを root の所有にして、他のユーザからはアクセス不可能としてしまえば、そこにファイルを置いたユーザはファイルを取り出せなくなるので、こういったことは起こりづらい。この問題を避けるには、全てのユーザからは書き込めないディレクトリ(例えば、/var/adm/sudo)をタイムスタンプディレクトリとして使うか、システムのスタートアップファイルで/var/run/sudo を適切な所有者(root)とアクセス権(0700)で作成すればよい。

sudo は、遠い未来の時刻になっているタイムスタンプのセットを受け付けない。current_time + 2 * TIMEOUT より先の時刻になっているタイムスタンプは無視され、sudo はログに記録を残して警告を出す。これにより、ユーザによるファイル譲渡が可能なシステム上で、ユーザが偽の日付でタイムスタンプを作成するのを防ぐ。

sudo は明示的に実行されたコマンドしかログに記録しない点に注意すること。ユーザが sudo su や sudo sh といったコマンドを実行した場合、シェルでそれ以降に実行されたコマンドはログに記録されず、sudo のアクセス制御も効かない。これはシェルエスケープを提供するコマンド(多くのエディタも含まれる)でも同じである。そのため、sudo を介してユーザがコマンドへアクセスするのを許可する場合は、有効な root のシェルをそのコマンドでうっかり与えてしまわないように注意しなければならない。

例

注意:以下の例は、適切な sudoers(5)エントリがあることを仮定している。

読み込み不可のディレクトリのファイルリストを取得する。

```
% sudo ls /usr/local/protected
```

~yazza の存在するファイルシステムが root でエクスポートされていないマシン上で、ユーザ yazza のホームディレクトリの中身をリストする。

% sudo -u yazza ls ~yazza

ファイル index.html をユーザ www として編集する。

% sudo -u www vi ~www/htdocs/index.html

マシンをシャットダウンする。

% sudo shutdown -r +15 "quick reboot"

/home パーティションにあるディレクトリのディスク使用量リストを作成する。cd とファイルリダイレクションが動作するように、サブシェルでコマンドを実行している点に注意すること。

% sudo sh -c "cd /home ; du -s * | sort -rn > USAGE"

環境変数

- sudo は、以下の環境変数を使用する。
- PATH - SECURE_PATH が設定されていると安全な値に設定される。
- SHELL - -s オプションで実行するシェルを決定するために使われる。
- USER - 対象となるユーザを設定する(-u オプションで指定されない限り root)。
- HOME - -s または-H オプション(または、sudo に--enable-shell-sets-home オプションが設定されていた)場合に、対象ユーザのホームディレクトリを設定する。
- SUDO_PROMPT - デフォルトのパスワードプロンプトとして使われる。
- SUDO_COMMAND - sudo の実行するコマンドに設定される。
- SUDO_USER - sudo を起動したユーザのログイン名に設定される。
- SUDO_UID - sudo を起動したユーザのユーザ ID に設定される。
- SUDO_GID - sudo を起動したユーザのグループ ID に設定される。
- SUDO_PS1 - この変数が設定されている場合、PS1 がこの変数の値に設定される。

ファイル

- /etc/sudoers - 誰が何を可能であるかのリスト。
- /var/run/sudo - タイムスタンプの含まれるディレクトリ。

警告

ユーザがシェルエスケープの可能なコマンドにアクセスできる場合、ユーザに root のシェルを入手させないための簡単な方法はない。

ユーザに sudo ALL の権限がある場合、そのユーザの設定に'!' 指定をしても、自分でプログラムを書け

ば root シェルを入手できてしまう。これを防ぐ簡単な方法はない。

シェルスクリプトを sudo で実行すると、カーネルのバグを突いてしまい、OS によっては setuid シェルスクリプトを危険なものにしてしまうかもしれない(使用している OS が/dev/fd/ディレクトリをサポートしている場合、setuid シェルスクリプトは一般的に安全である)。

関連項目

stat(2)、login_cap(3)、sudoers(5)、passwd(5)、visudo(8)、grep(1)、su(1)

6.1.3.3 新しいグループにログインする (newgrp)

書式

```
newgrp [-] [group]
```

説明

newgrp はログインセッション中に現在のグループ ID を変更するために用いられる。オプションとして-フラグを与えた場合は、新たにログインしたのと同じ様に環境が再初期化される。そうでない場合は、現在の作業ディレクトリを含めて、現在の環境は変化しない。

newgrp は現在の実グループ ID を、指定したグループに(グループ名を指定しなかった場合は /etc/passwd に記載されたデフォルトのグループに)変更する。ユーザにはパスワードがなくグループにはある場合、あるいはユーザがグループのメンバーではなくグループにパスワードがある場合には、そのユーザはパスワードの入力を求められる。グループのパスワードが設定されておらず、かつユーザがグループのメンバーでない場合は、アクセスは拒否される。

ファイル

- /etc/passwd - ユーザアカウント情報
- /etc/group - グループ情報

関連項目

id(1)、login(1)、su(1)

6.1.3.4 別のグループ ID でコマンドを実行する (sg)

書式

sg [-] [group [[-c] command]]

説明

sg コマンドは newgrp と同様に動作するが、コマンドを受け付ける。このコマンドは Bourne シェルで実行される。コマンドが複数の単語からなる場合は、sg の実行元となるであろうシェルのほとんどにおいて、これらをクォートする必要があるだろう。newgrp と sg のもう一つの違いは、特定のシェルが newgrp を特別に扱う点にある。このようなシェルは、自分自身を newgrp が生成した新しい実体と置き換える。このようなことは sg では起きないので、sg コマンドから戻った際には以前のグループ ID に戻る。

ファイル

- /etc/passwd - ユーザアカウント情報
- /etc/group - グループ情報

関連項目

id(1)、login(1)、su(1)

6.1.4. アクセス制御属性管理

6.1.4.1 ファイルのアクセス権を変更する (chmod)

書式

chmod [options] mode file...

POSIX オプション: [-R]
GNU オプション (簡略形式): [-cfvR] [--reference=file] [--help] [--version] [--]

説明

chmod コマンドは指定したそれぞれの file のアクセス権を mode により変更する。変更方法をシンボルで表現するか、もしくは変更後のアクセス権を表すビットパターンを 8 進数で表現したもののいずれかを使うことができる。

シンボルモードでアクセス権を変更する場合の引数の書式は '[ugoa...][[+-=][rwxXstugo...]...][,...]' であ

る。

シンボルモードで引数にアクセス権の変更を複数指定する場合にはコンマで区切る。シンボルモードでアクセス権を変更するそれぞれの表現は 0 個もしくは複数の文字 'ugoa' で始まる;これらがファイルに対するどのユーザーのアクセス権を変更するかを決める:ファイルの所有者(u)、ファイルと同じグループに属しているが所有者ではない(訳注:以後グループと表記)(g)、所有者でもなく、そのファイルと同じグループにも属していない(訳注:以後その他と表記)(o)、もしくは全ユーザー(a)を表している。つまり 'a' は 'ugo' と同じである。もしこれらのいずれも指定されない場合、'a' が指定されたのと同じ結果となるが、umask に設定されたビット位置は変化しない。

'+' 演算子は各々のファイルの現状のアクセス権に、選択したアクセス権を加える; '-' は削除する;そして '=' は指定したアクセス権だけがそのファイルのアクセス権となる。

文字 'rwxXstugo' は影響を与えるユーザーに対する新しいアクセス権を選択する:読取り(r)、書き込み(w)、実行(またはディレクトリに対するアクセス)(x)、ファイルがディレクトリかもしくはあるユーザーに対してすでに実行アクセス権が設定されている場合のみ実行(X)、実行時にユーザーもしくはグループ ID を設定する(s)、sticky bit(t)、そのファイルの所有者での現在のアクセス権は(u)、そのファイルのグループでの現在のアクセス権は(g)、そのファイルのその他のユーザーでの現在のアクセス権は(o)で表される。(従って、'chmod g-s file' は set-group-ID(sgid)ビットを削除する。 'chmod ug+s file' は suid そして sgid ビットの両方を設定するが、 'chmod o+s file' は何もしない。)

POSIX では 'sticky bit' について記述していない。その名前は本来の意味から来ている:プログラムコードをスワップ上に維持する。最近は、ディレクトリに設定されている場合、ファイルの所有者とディレクトリの所有者だけがそのディレクトリからファイルを削除できることを意味する。(全ユーザーが書き込みアクセス権を持つ/tmp のようなディレクトリでこれはごく普通に使われている。)

chmod コマンドがシンボリックリンクのアクセス権を変更することは決してない。なぜなら、chmod システムコールはシンボリックリンクのアクセス権を変更することができないからである。シンボリックリンクのアクセス権は決して使われることがないため、このことは問題ではない。しかし、chmod コマンドは引数にシンボリックリンクが指定された場合、各々についてそれが指しているファイルのアクセス権を変更する。それに対して、chmod コマンドは再帰的にディレクトリを移動しながら処理している時に見つけたシンボリックリンクは無視する。

POSIX オプション

- -R
  ディレクトリやそこに含まれるもののアクセス権を再帰的に変更する。
- -c、--changes
  実際にアクセス権の変更があった file それぞれについての動作を詳細に表示する。

- -f、--silent、--quiet

  アクセス権を変更できなかったファイルについてのエラーメッセージを出力しない。

- -v、--verbose

  全ての file について変更した、もしくはしなかったという動作を詳細に表示する。

- -R、--recursive

  ディレクトリやそこに含まれるもののアクセス権を再帰的に変更する。

- --reference=file

  明示的に mode を指示する代わりに、参照用に指定した file のグループを使う。

GNU 標準オプション

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

- --

  オプションリストを終了する。

数値モード

ファイルのアクセス権は内部的に16ビットの整数で保持されている。シンボルでアクセス権を指定する代わりに、これから変更するアクセス権の内部表現に対応した 8 進数(基数 8)を使って指定することができる。この数値は常に 8 進数として扱われる;C 言語のように数値の先頭に 0 を付加する必要はない。mode に 0055 と指定するのと 55 と指定するのは同じである。数値モードによる指定は、シンボルモードで指定するより大抵の場合短くすむが、アクセス権を絶対値で指定する。mode の各数字により、以下で示すようなアクセス権のグループを選択する。左側の省略された数値は 0 として扱われる。アクセス権を組み合わせるには、各数字の OR をとるか各数字を合計すればよい。

特別なアクセス権

- 1000 プログラムコードをスワップに維持
- 2000 実行時にグループ ID を設定
- 4000 実行時にユーザーID を設定

同じ UID のユーザー(u+)

- 100 実行
- 200 書き込み
- 400 読み込み

同じグループ(g+)

- 10 実行
- 20 書き込み
- 40 読み込み

その他のユーザー(o+)

- 1 実行
- 2 書き込み
- 4 読み込み

環境変数

変数 LANG、LC_ALL、LC_CTYPE、LC_MESSAGES が通常の意味を持つ。

準拠

POSIX 1003.2 では-R オプションのみが必須である。その他のオプションを使用すると互換性がないかもしれない。この標準では't' の許可ビットについて説明していない。この標準では chmod が suid や sgid ビットのクリアもしくは設定の拒絶での一貫性を維持するべきかどうかについてをとり決めていない。たとえば、すべての実行ビットがクリアされる場合に chmod が 's' ビットをどうするかまったくとり決めていない。

非標準モード

ここまでのところでディレクトリに対する 't' ビットの使い方を説明した。いろいろなシステムでは、他の意味のないモードビットの組合せに特別な意味を与えている。特に、Linux は SystemV 系(System V Interface Definition (SVID) Version 3 を参照)に倣って、グループ実行許可を持たないファイルの sgid ビットに、そのファイルが強制ロック(mandatory locking)の対象であることを示させている。詳細については /usr/src/linux/Documentation/mandatory.txt ファイルを参照。

6.1.4.2 ファイルの所有者とグループを変更する (chown)

書式

```
chown [オプション]... OWNER[.[GROUP]] FILE...
chown [オプション]... .GROUP FILE...
chown [オプション]... --reference=RFILE FILE...
```

説明

このマニュアルは GNU 版 chown コマンドについての記述である。chown コマンドは、指定されたそれぞれのファイルのユーザおよびグループ、もしくはいずれかの所有権を、オプションではない最初の引数に従って、以下のように変更する。ユーザ名(もしくは数値の user ID)のみを指定した場合、それぞれのファイルの所有者は指定したユーザになり、グループは変わらない。ユーザ名に続けてコロンもしくはドットとグループ名(もしくは数値の group ID)を間にスペースを入れずに指定した場合、同じようにそれらのファイルのグループ所有権も指定したグループに変わる。ユーザ名に続いてコロンもしくはドットがあるのにグループ名が無い場合、ファイルの所有権はそのユーザになり、ファイルのグループはそのユーザのログイングループに変更される。コロンもしくはドットとグループは指定されているがユーザ名が無い場合、ファイルのグループのみが変更される。この場合、chown コマンドは chgrp コマンドと同じ働きをする。

オプション

それぞれの FILE の所有者およびグループもしくはいずれかを、OWNER および GROUP もしくはいずれかに変更する。

- -c、--changes
  実際に変更があった場合の動作を詳細に表示する。
- --dereference
  シンボリックリンクそれ自身ではなく、指している先を変更する。
- -h、--no-dereference
  指している先ではなく、シンボリックリンクそれ自身を変更する。(シンボリックリンクの所有権を変更できるシステムの場合のみ有効)
- -f、--silent、--quiet
  ほとんどのエラーメッセージの出力を抑える。
- --reference=RFILE
  明示的に OWNER.GROUP 値を指示する代わりに、指定したファイル(RFILE)のユーザ、グループを使う。
- -R、--recursive
  ファイルやディレクトリの所有権を再帰的に変更する。
- -v、--verbose
  実行した内容を表示する。
- --help
  使用方法を表示して正常終了する。
- --version
  バージョン情報を出力して正常終了する。

所有者は指定しないと変更されない。グループは指定しないと変更されないが、コロンもしくはドットで指定された場合は、ログイングループに変更される。

関連項目

chown の完全なドキュメントは info マニュアルとしてメンテナンスされている。info と chown が正しくインストールされていれば、info chown コマンドで完全なマニュアルを参照することができる。

6.1.4.3 ファイルのグループ所有権を変更する (chgrp)

書式

```
chgrp [options] group file...

POSIX オプション: [-R]
GNU オプション(簡略形式): [-cfvR] [--dereference] [--reference=file] [--help] [--version]
[--]
```

説明

chgrp コマンドは指定されたそれぞれの file のグループ所有権を group に変更する。その指定方法としてグループ名、数値でのグループ ID、もしくは参照用のファイルを使うことができる。

POSIX オプション

- -R

  ディレクトリやそこに含まれるもののグループ所有権を再帰的に変更する。(エラーが起こっても処理を続ける。)

GNU オプション

- -c、--changes

  実際にグループの変更があった file それぞれについての動作を詳細に表示する。

- --dereference

  シンボリックリンク自身ではなく、それが指している先を変更する。

- -f、--silent、--quiet

  グループを変更できなかったファイルについてのエラーメッセージを出力しない。

- -h、--no-dereference

  指している先ではなく、シンボリックリンクそれ自身を変更する。これがデフォルトである。lchown(2)システムコールが提供されていない場合、chown は失敗する。再帰的に辿っている

ときにシンボリックリンクが見つかり、かつ--verbose が指定されていない場合、エラーメッセージは表示されない。

- -v、--verbose

  全ての file について変更した(もしくはしなかった)という動作を詳細に表示する。

- -R、--recursive

  ディレクトリやそこに含まれるもののグループ所有権を再帰的に変更する。

- --reference=file

  指定されたファイルとディレクトリを file と同じグループに変更する。

GNU 標準オプション

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

- --

  オプションリストを終了する。

環境変数

変数 LANG、LC_ALL、LC_CTYPE、LC_MESSAGES が通常の意味を持つ。

準拠

POSIX 1003.2 では-R オプションのみを必要としている。その他のオプションを使用すると互換性がないかもしれない。

注意

chown(2)が set-uid、set-gid のビットをリセットするようなシステムにおいても、これらは保存される。

6.1.4.4 ファイル生成マスクを設定する (umask)

書式

```
umask [-p] [-S] [mode]
```

説明

ユーザーのファイル生成マスクに mode を設定します。mode が数字で始まる場合には、これは 10 進数と

解釈されます。それ以外の場合には、chmod(1)に指定するのと同様のシンボリックリンクなモードマスクと解釈されます。Mode が省略されると、現在のマスクの値が出力されます。-S オプションを指定すると、マスクはシンボリックな形式で表示されます。デフォルトの出力は 10 進の数値です。-p オプションが指定され、かつ mode が省略された場合、入力として再利用できる形式で出力が行われます。モードが正常に変更できた場合や、mode 引数が全く与えられなかった場合には、返却されたステータスは 0 となります。それ以外の場合は偽となります。

6.2. 管理者コマンド

6.2.1. ユーザ/グループ管理

6.2.1.1 新規ユーザの作成、および新規ユーザのデフォルト情報の変更 (adduser、useradd)

書式

```
( adduser | useradd ) [-c comment] [-d home_dir]
  [-e expire_date] [-f inactive_time]
  [-g initial_group] [-G group[,...]]
  [-m [-k skeleton_dir]] [-p passwd]
[-s shell] [-u uid [ -o]] login

( adduser | useradd ) -D [-g default_group] [-b default_home]
[-f default_inactive] [-e default_expire_date]
[-s default_shell]
```

説明

新規ユーザの作成

-D オプションなしで実行された場合、useradd コマンドは、コマンドラインで与えられた値とデフォルトの設定値をもとに、新規ユーザのアカウントを作成する。コマンドラインのオプションに応じて、新規ユーザのアカウントが必要なシステムファイルに追加され、ホームディレクトリが作られ、設定ファイルがコピーされる。useradd コマンドのオプションは次のとおりである。

- -c comment

  パスワードファイルに追加する新規ユーザのコメントフィールド。

- -d home_dir

  新規ユーザのログイン時のディレクトリは、home_dir とする。デフォルトでは、login を default_home に付け加えたものがログイン時のディレクトリとなる。

- -e expire_date

  ユーザアカウントが無効となる日付。日付は YYYY-MM-DD の形式である。

- -f inactive_days

  パスワードがの使用期限が切れてから、このオプションで与えた日数経過するとアカウントは永久に使用不能となる。値として 0 を指定すると、パスワードが失効した直後にアカウントは使用不能となり、-1 を指定すると、この機能は無効となる。デフォルト値は-1 である。

- -g initial_group

ユーザの属する主グループのグループ名またはグループ ID。グループ名はすでに存在するものでなければならない。グループ ID は、すでに存在するグループに対応するものでなければならない。デフォルトのグループ ID は 1 である。

- -G group,[...]

  ユーザの属する補助グループのリスト。グループはコンマで区切り、空白を含めてはいけない。これらのグループは、-g オプションと同様に、すでに存在するものでなければならない。デフォルトでは、ユーザは主グループのみに属する。

- -m

  ホームディレクトリが存在しない場合には、ホームディレクトリを作成する。-k オプションを同時に指定すると skeleton_dir 以下のファイルが、指定しないと/etc/skel 以下のファイルが、ホームディレクトリにコピーされる。skeleton_dir または/etc/skel に含まれるすべてのディレクトリも、ホームディレクトリに作られる。-k オプションは、-m オプションとともに使われる場合のみ有効である。デフォルトでは、ホームディレクトリを作らず、ファイルのコピーもしない。

- -p passwd

  crypt(3)によって暗号化されたパスワード。デフォルトでは、アカウントは使えない状態となる。

- -s shell

  ユーザのログインシェル名。デフォルトではこのフィールドは空白となり、システムがデフォルトのログインシェルを選ぶ。

- -u ui

  ユーザ ID。-o オプションが同時に指定されない場合は、他と重なってはならない。数値は非負の値でなければならない。デフォルトでは、99 より大きく、すでに存在するどのユーザよりも大きい数のうち、最小の値が使われる。0 から 99 までの値は大抵、システムアカウント用として予約されている。

デフォルト値の変更

-D オプションを指定すると、useradd は現在のデフォルト値を表示するか、またはオプションで与えられた値に応じてデフォルト値を変更する。使用可能なオプションは次のとおりである。

- -b default_home

  新規ユーザのホームディレクトリへのパス。新規ユーザアカウントを作成する際に-d オプションを指定しない場合、default_home の後にユーザ名を付け加えたものが新規ディレクトリ名として使われる。

- -e default_expire_date

  ユーザアカウントが無効となる日付。

- -f default_inactive

  パスワードの使用期限が切れてからアカウントが使用不能となるまでの日数。

- -g default_group

新規ユーザの属する主グループのグループ名またはグループ ID。グループ名はすでに存在するものでなければならない。グループ ID は、すでに存在するグループに対応するものでなければならない。

- -s default_shell

  新規ユーザのログインシェル。指定されたプログラムが、今後作られるすべてのユーザアカウントに適用される。

オプションを指定しない場合、useradd は現在のデフォルト値を表示する。

注意

/etc/skel ディレクトリにデフォルトのユーザファイルを置く作業はシステム管理者の責任である。

警告

NIS のグループにユーザを加えてはならない。これは必ず NIS サーバ上で行うこと。

ファイル

- /etc/passwd - ユーザアカウント情報
- /etc/shadow - shadow されたユーザアカウント情報
- /etc/group - グループ情報
- /etc/default/useradd - デフォルト値の情報
- /etc/skel - ファイルの雛形が置かれるディレクトリ

関連項目

chfn(1)、chsh(1)、passwd(1)、crypt(3)、groupadd(8)、groupdel(8)、groupmod(8)、userdel(8)、usermod(8)

6.2.1.2 ユーザのアカウント及び関連するファイルを削除する (userdel)

書式

```
userdel [-r] login
```

説明

userdel コマンドはシステムのアカウントファイルに変更を施し、ユーザ login に属する全てのエントリーを削除する。削除されるユーザは存在していなければならない。

オプション

- -r

  ユーザのホームディレクトリ中のファイルをホームディレクトリ自体とともに消去する。またユーザのメールスプールも消去する。他のファイルシステム上にあるファイルは手作業で探し出して除去しなくてはならない。

ファイル

- /etc/passwd - ユーザのアカウントに関する情報
- /etc/shadow - 安全な、ユーザのアカウントに関する情報
- /etc/group - グループに関する情報

警告

userdel は消去されるユーザが現在ログインしている場合はそのアカウントを消去する事を許可しない。その時は消去しようとしているアカウントに属する実行中のプロセス全てを kill しなくてはならない。NIS のクライアントからはいかなる NIS 所有のアカウントも消去する事は出来ない。消去する場合には NIS のサーバから行わなくてはならない。

関連項目

chfn(1)、chsh(1)、passwd(1)、groupadd(8)、groupdel(8)、groupmod(8)、useradd(8)、usermod(8)

6.2.1.3 ユーザアカウントを変更する (usermod)

書式

```
usermod [-c comment] [-d home_dir [ -m]]
    [-e expire_date] [-f inactive_time]
    [-g initial_group] [-G group[,...]]
    [-l login_name] [-p passwd]
    [-s shell] [-u uid [ -o]] [-L|-U] login
```

説明

usermod コマンドはコマンドライン上での指示にしたがってシステムのアカウントファイルを変更する。

オプション

- -c comment

パスワードファイルの新しいコメント欄の内容通常は chfn(1)ユーティリティを用いて変更される。

- -d home_dir

  新しいログインディレクトリ。-m オプションを用いた場合は現在のホームディレクトリの中身が新しいホームディレクトリに移動される。もし存在しない場合は新たに作られる。

- -e expire_date

  アカウントが使用不能になる日付。日付は YYYY-MM-DD という書式で指定する(YYYY、MM、DD はそれぞれ年、月、日を表す数字)。

- -f inactive_days

  パスワードの使用期限が切れてからアカウントが永久に使用不能になるまでの日数。0とすると、パスワードの期限が切れると同時にこのアカウントは使用不可能になる。-1 とするとこの機能が働かなくなる。デフォルト値は-1。

- -g initial_group

  ログイン時の新しいグループ名または ID。このグループ名は既に存在していなくてはならない。また、グループ番号は既存のグループを参照していなくてはならない。デフォルトのグループ番号は 1 である。

- -G group,[...]

  ユーザが属す、副グループのリスト。グループはコンマを用いて区切り、間に空白文字を入れてはならない。指定できるグループには-g オプションを用いる場合と同様の制限がある。新しいリストにないグループのメンバーになっている場合は、そのグループから削除される。

- -l login_name

  ユーザのログイン名を login から login_name に変更する。他は何も変更しない。特に、新しいログイン名に合わせてホームディレクトリ名を変更しなくてはならないだろう。

- -p passwd

  crypt(3)の返り値である暗号化されたパスワード。

- -s shell

  新しいログインシェルの名前。この欄を空白にした場合はシステムがデフォルトのログインシェルを選択する。

- -u uid

  ユーザの ID 番号。この番号は他と重複していてはならない。-o オプションを用いた際はこの限りではない。また、非負値でなくてはならない。0 から 99 迄の値は大抵システム用のアカウントのために予約されている。ホームディレクトリ以下の、そのユーザ所有の全てのファイルのユーザ ID は、自動的に新しい値に変更される。ホームディレクトリ以下にないファイルは手作業で変更しなくてはならない。

- -L

  ユーザのパスワードをロックする。これは暗号化されたパスワードの先頭に'!'を追加し、事実

上パスワードを無効にする。このオプションは-p または-U と同時に用いることはできない。

- -U

  ユーザのパスワードをアンロックする。これは暗号化されたパスワードの先頭の'!' を取り除く。このオプションは-p または-L と同時に用いることはできない。

警告

usermod は現在ログインしているユーザの名前を変更する事は出来ない。このコマンドを使用してユーザの ID 番号を変更する際は、指定したユーザのプロセスが一つも実行されていない事を確認してからでなくてはならない。crontab ファイルの所有者は手作業で変更しなくてはならない。また、at ジョブの所有者も手作業で変更する必要がある。NIS に関する作業は NIS サーバ上で行なわなくてはならない。

ファイル

- /etc/passwd - ユーザアカウントの情報
- /etc/shadow - 安全なユーザアカウント情報
- /etc/group - グループ情報

関連項目

chfn(1)、chsh(1)、passwd(1)、crypt(3)、groupadd(8)、groupdel(8)、groupmod(8)、useradd(8)、userdel(8)

6.2.1.4 新しいグループを作成する (groupadd)

書式

```
groupadd [-g gid [-o]] group
```

説明

groupadd コマンドはコマンドライン上で指定された値及びシステムのディフォルト値を用いて新しいグループを作成する。新しいグループは必要となった際にシステムファイルに記入される。

オプション

- -g gid

  新規グループの ID 番号。この値は、-o オプションを用いない限りは、他と重複してはならない。また、非負値でなくてはならない。ディフォルトは、99 より大きく且つ他の全ての既存グループの ID よりも大きなものの中で最小のものである。0 から 99 迄の値は大抵システム用のアカウントに予約されている。

ファイル

- /etc/group - グループのアカウント情報
- /etc/gshadow - グループの安全なアカウント情報

関連項目

chfn(1)、chsh(1)、passwd(1)、groupdel(8)、groupmod(8)、useradd(8)、userdel(8)、usermod(8)

6.2.1.5 グループを消去する (groupdel)

書式

groupdel group

説明

groupdel コマンドは、システムのアカウントファイルを変更し、group に属する全てのエントリーを削除する。指定されたグループは存在していなくてはならない。

全てのファイルシステム中に指定したグループの ID を持つファイルが残っていないことを、手作業で確認しなくてはならない。

警告

存在するユーザの主グループを削除してはならない。グループを削除する前にそのユーザを削除しなくてはならない。

ファイル

- /etc/group - グループの情報
- /etc/gshadow - グループの安全な情報

関連項目

chfn(1)、chsh(1)、passwd(1)、groupadd(8)、groupmod(8)、useradd(8)、userdel(8)、usermod(8)

6.2.1.6 グループに変更を加える（groupmod）

書式

groupmod [-g gid [-o]] [-n group_name] group

説明

groupmod コマンドは、コマンドライン上で指定した変更事項に即してシステムのアカウントファイルを改変する。

オプション

- -g gid

  変更を受けるグループの ID 番号。この値は、-o オプションを用いる場合以外は、他と重複してはならない。また、非負値でなくてはならない。ディフォルトでは、99 より大きく且つ他の全ての既存グループの ID よりも大きなものの中で最小のものである。変更前のグループ ID を持つ全てのファイルは、手作業で新しいグループ ID へと変更しなくてはならない。
- -n group_name

  グループの名前が group から group_name に変更される。

ファイル

- /etc/group - グループの情報
- /etc/gshadow - グループの安全な情報

関連項目

chfn(1)、chsh(1)、passwd(1)、groupadd(8)、groupdel(8)、useradd(8)、userdel(8)、usermod(8)

6.2.1.7 ユーザ情報の変更や新規ユーザ作成をまとめて行う（newusers）

書式

newusers [new_users]

説明

newusers は、ユーザ名と平文パスワードの対を記したファイルを読み、その情報をもとに、既存のユーザ情報の変更や、新規ユーザの作成を行う。このファイルの書式は、下記の点を除き、標準的なパスワー

ドファイル(passwd(5)参照)と同じである。

- pw_passwd このフィールドが暗号化され、暗号化パスワードとなる。
- pw_age ユーザがすでに存在する場合、パスワードを隠すため、このフィールドは無視される。
- pw_gid このフィールドに既存のグループを指定した場合は、ユーザはそのグループに加えられる。存在しないグループ ID を指定した場合は、そのグループ ID をもつ新たなグループが作られる。
- pw_dir このフィールドで指定したディレクトリがすでに存在するかチェックし、もし存在しなければ、新たにディレクトリが作られる。また、このディレクトリは、新規ユーザ、あるいは情報が変更されたユーザが所有者となる。

このコマンドは、多くのアカウントが一度に変更されるような大きなシステム環境で使うためのものである。

警告

入力ファイルには、生のパスワードが含まれるため、扱いに注意すること。

関連項目

passwd(1)、useradd(8)

6.2.2. パスワード管理

6.2.2.1 パスワードファイルをバッチ処理で更新する (chpasswd)

書式

chpasswd [-e]

説明

chpasswd コマンドは標準入力からユーザ名とパスワードの組が記されているファイルを読み込み、その情報を用いて既存のユーザ群のパスワード情報を更新する。-e オプションがない場合は、パスワードは平文と見なされる。-e オプションがある場合は、パスワードは暗号化されていると見なされる。各行は

user_name:password

という書式である。指定したユーザは既に存在していなくてはならない。与えられたパスワードは必要に応じて暗号化され、パスワードの有効期限が存在するならそれも更新される。

このコマンドは、同時に大量のアカウントを作成する様な大規模なシステム環境で使用することを意図している。

警告

入力ファイルに暗号化されていないパスワードが記されている場合は、プロテクトをかけておかなくてはならない。

関連項目

passwd(1)、useradd(8)、newusers(8)

6.2.3. アクセス制御属性管理

6.2.3.1 ext2/ext3 ファイルシステムのアトリビュートを変更する (chattr)

書式

```
chattr [ -RV ] [ -v version ] [ mode ] files...
```

説明

chattr は Linux 第 2 拡張ファイルシステム(ext2fs)上にあるファイルの属性(attribute)を変更する。

mode の記法フォーマットは+-=[ASacDdIijsTtu]である。

+オペレータを使用すると、選択した属性がファイルに付加される。-オペレータを使用すると解除される。=オペレータを使用すると、ファイルは指定した属性だけを持つことになる。

「acdijsuADST」は指定できる属性である。(a)追加のみ許可(append only)、(c)圧縮(compressed)、(d)ダンプしない(no dump)、(i)変更不可 (immutable)、(j)データのジャーナリング(data journalling)、(s)安全な削除(secure deletion)、(t)末尾マージをしない(no tail-merging)、 (u)復活可能(undeletable)、(A)atime を更新しない、(D)ディレクトリの同期更新(synchronous directory updates)、(S)同期更新(synchronousupdates)、(T)ディレクトリ階層のトップ。

オプション

- -R

  再帰的にディレクトリやその中身の属性を変更する。ディレクトリを順番に訪れている途中でシンボリックリンクに出会った場合は、これを無視する。
- -V

属性の変更を詳細に表示する。

- -v version

  ファイルシステムのバージョン/世代を設定する。

属性

'A' 属性が設定されているファイルは、アクセスされても atime レコードが変更されない。これはラップトップシステムのディスク I/O をある程度軽減する。

'a' 属性が設定されているファイルは、書き込みの際に追加モードでしかオープンできない。スーパーユーザーまたは CAP_LINUX_IMMUTABLE ケーパビリティ(capability)を持つプロセスだけが、この属性を設定・解除できる。

'c' 属性が設定されているファイルは、ディスク上に置かれるときカーネルによって自動的に圧縮される。このファイルを読み出すと、伸長されたデータが返ってくる。このファイルに書き込むと、ディスク上に保存する前にデータが圧縮される。

'D' 属性が設定されているディレクトリは、変更されると同時にディスクにもその内容が書き込まれる。これは 'dirsync' マウントオプションを、それらのファイルだけに適用するのと同じことである。

'd' 属性が設定されているファイルは、dump(8)プログラムを起動した際にバックアップされない。

'E' 属性は実験的な圧縮パッチが用いるもので、圧縮ファイルに圧縮エラーがあることを示す。この属性は chattr(1)によって設定したり解除したりすることはできないが、lsattr(1)を用いて表示させることはできる。

'I' 属性は htree コードによって用いられ、ディレクトリがハッシュツリーを使って索引を付けられることを示す。この属性は chattr(1)によって設定したり解除したりすることはできないが、lsattr(1)を用いて表示させることはできる。

'i' 属性が設定されているファイルは、変更することができない。すなわち、削除、名前の変更、このファイルを指すリンクの作成、このファイルに対するデータの書き込みが禁止される。スーパーユーザーまたは CAP_LINUX_IMMUTABLE ケーパビリティを持つプロセスだけが、この属性を設定・解除できる。

'j' 属性が設定されているファイルは、ファイルシステムが"data=ordered"または"data=writeback"オプションでマウントされている場合、ファイルそのものに書き出される前に、ext3 のジャーナルにデータが書き出される。ファイルシステムが"data=journal"オプションでマウントされている場合、全てのファイルデータは既にジャーナルに記録されているので、この属性は何も影響しない。スーパーユーザーまたは CAP_SYS_RESOURCE ケーパビリティを持つプロセスだけが、この属性を設定・解除できる。

's' 属性が設定されているファイルが削除されると、割り当てられていたブロックの中身が 0 にされ、ディ

スクに書き戻される。

‘S’ 属性が設定されているファイルが変更されると、変更は同期的にディスクに書き込まれる(すなわち変更が直ちにディスク上に反映される)。これはmountオプションの ‘sync’ をそれらのファイルだけに適用するのと同じことである。

‘T’ 属性が設定されているディレクトリは、(Linux2.5.46 以降のシステムで使われる)Orlov ブロックアロケータのためのディレクトリ階層のトップとして扱われる。

‘t’ 属性が設定されているファイルは、そのファイルが他のファイルとマージされたときに部分的ブロックのフラグメントを末尾に作らない(末尾マージ (tail-merging)をサポートしているファイルシステムの場合)。これは LILO のような、ファイルシステムを直接読みにいき、かつ末尾マージされたファイルのことを理解しないようなアプリケーションに対して必要となる。注:この man ページを書いている時点で、ext2 と ext3 ファイルシステムは(まだ非常に実験的なパッチ以外では)末尾マージをサポートしていない。

‘u’ 属性が設定されているファイルを削除すると、その内容は保護される。これにより、ユーザがファイルを復活させることが可能になる。

‘X’ 属性は実験的な圧縮パッチによって用いられ、圧縮ファイルの内容が圧縮状態のまま直接アクセス可能であることを示す。この属性は現在のところ chattr(1)によって設定したり解除したりすることはできないが、lsattr(1)を用いて表示させることはできる。

‘Z’ 属性は実験的な圧縮パッチによって用いられ、圧縮ファイルが汚染されている(dirty)ことを示す。この属性は chattr(1)によって設定したり解除したりすることはできないが、lsattr(1)を用いて表示させることはできる。

バグと制限

‘c’、 ’s’、‘u’ 属性への対応は、現在主流になっている Linux カーネルで実装されている ext2 と ext3 ファイルシステムに含まれていない。これらの属性は将来の ext2 と ext3 に実装されるかもしれない。

‘j’ オプションは、ext3 でマウントされたファイルシステムについてのみ有効である。

‘D’ オプションは Linux カーネル 2.5.19 以降でのみ利用できる。

入手方法

chattr は e2fsprogs パッケージの一部であり、 http://e2fsprogs.sourceforge.net から入手できる。

関連項目

lsattr(1)

6.2.4. 監査

6.2.4.1 Linux システムロギングユーティリティ (sysklogd)

書式

```
syslogd [ -a socket ] [ -d ] [ -f config file ] [ -h ] [ -l hostlist ] [ -m interval ] [ -n ]
[ -p socket ] [ -r ] [ -s domainlist ] [ -v ]
```

説明

sysklogd はシステムロギングとカーネルメッセージの確保という二つの機能を提供するユーティリティである。このユーティリティではインターネットとUNIXドメインの両方のソケットが利用可能なので、ローカルとリモートの両方で記録可能である。

このシステムの記録を提供する syslogd(8)は BSD のソースコードに由来しているものである。カーネルロギング機能は klogd(8)ユーティリティによって提供され、スタンドアロン形式、syslogd クライアントのどちらの形式でも動作する。

syslogd は最近の多くのプログラムにロギングの手段を提供する。記録されるメッセージは少なくとも時間の情報とホスト名を持ち、通常はそれらに加えてプログラム名のフィールドも持っている。しかしながら記録される内容の信頼性についてはそれぞれのプログラムに依存する。

syslogd のソースコードは激しく変更されながらも二つの特徴についてはそれを維持しつづけている。まず第一に、その基本となる、一般的な BSD らしい挙動への追随を保証することの系統的な意図の維持である。第二の重要な特徴は、この版の syslogd が標準ライブラリに含まれる syslog と透過的に相互に影響しあっていることである。もし標準の共有ライブラリがリンクされたバイナリコードが正常に機能しないならば、作者達にその異様な挙動の例を送って欲しい。

起動時にその主設定ファイルである/etc/syslog.conf かまたは代替として-f オプションであたえられる名前のファイルが読み込まれる。その各行のうち、シャープ記号(‘‘#’’)ではじまるものと空行は無視する。その行の解析においてエラーを生じた行もまた無視する。

オプション

- -a socket

  この引数を使って、syslogd が listen する追加のソケットを指定できる。これはデーモンを

chroot()環境で動作させようとする時に必要である。追加のソケットは 19 個まで指定できる。もしもっと多くのソケットが必要なら、syslogd.c ファイルの MAXFUNIX シンボルの値を増やす必要がある。chroot() デーモンの例としては OpenBSD の人が記した http://www.psionic.com/papers/dns.html がある。

- -d

  デバッグモードを有効にする。このときデーモンは自身をバックグラウンドに推移させるための fork(2)を使用せずにフォアグラウンドに留まり、豊富なデバッグ情報を現在有効な tty へ出力する。このマニュアルページのデバッギングの章に詳細な解説があるので参照すること。

- -f config file

  既定値の/etc/syslog.conf ファイルの代替として、指示された名前のファイルを使用する。

- -h

  syslogd の既定値の設定では、リモートのホストから受信したメッセージをそれ以上転送することはない。コマンドラインからこのスイッチを使用するとデーモンはリモートホストから受信したメッセージをさらに別に定義されるホストへ転送する。

- -l hostlist

  記録するホストとして、FQDN ではなく単純にホスト名のみを指定する。コロン(‘ ‘:’ ’)を区切りに用いて複数のホストを指定することもできる。

- -m interval

  syslogd に定期的にタイムスタンプを記録させる。二行の-- MARK --を記録する間隔 interval の既定値は 20 分であり、このオプションで変更可能である。interval を 0 にすると、-- MARK --行を全く出力しない。

- -n

  自動的なバックグラウンドへの移行を抑止する。これは syslogd が init(8)により起動および制御される場合にのみ必要である。

- -p socket

  /dev/log の代りに指定した UNIX ドメインソケットを利用する。

- -r

  このオプションで、ネットワーク上でインターネットドメインソケットにより syslog サービス(services(5)を参照せよ)を使用してメッセージを受信する機能が有効になる。既定値ではネットワークからのいかなるメッセージも受信しない。

  このオプションは sysklogd パッケージの version1.3 で導入された。既定値の挙動がより以前の版の挙動の反対となっていて(ネットワーク越しに受信するには)このオプションを利用しなければならないという点にどうか注意していただきたい。

- -s domainlist

  記録する際に剥ぎ取るドメイン名を指定する。コロン(‘ ‘:’ ’)を区切りに用いて複数のドメインを指定することもできる。ドメインの一部ではなく、ドメイン全体を指定することに注意すること。

例えば、-snorth.de と指定して、ログを記録するホスト名が satu.infodrom.north.de だった場合、ドメインは剥ぎ取られない。このような場合、-s north.de:infodrom.north.de のように二つのドメインを指定しなければならない。

- -v

  バージョンを出力し、終了する。

シグナルの処理

syslogd はシグナルに反応する。syslogd に簡単にシグナルを送るには次のように実行すればよい:

```
kill -SIGNAL `cat /var/run/syslogd.pid
```

- SIGHUP

  このシグナルは syslogd に再初期化を指示する。全ての開いていたファイルを閉じ、設定ファイル(既定値では、/etc/syslog.conf)を再度読み込んで、syslog(3)の機能を再び有効にする。
- SIGTERM

  syslogd は終了する。
- SIGINT、SIGQUIT

  デバッグモードが有効である場合は無視するが、そうでなければ syslogd は終了する。
- SIGUSR1

  デバッグモードのオン/オフを切替える。この動作は syslogd がその起動時に-d オプションが指示されている場合にのみ有効。
- SIGCHLD

  メッセージの一斉通知のために、子プロセスがあればその終了を待つ。

設定ファイル文法の差異について

syslogd の設定ファイル文法はオリジナルの BSD のソースコードとは微妙に差異がある。オリジナルでは、指示されたものとそれ以上の優先順位をもつメッセージは全てログファイルに記録される。

たとえば、下記の例は daemon facility を使用する「全て(debug は最低の優先順位なので、それ以上の全てが適合するわけである)」のデーモンからの出力が/usr/adm/daemons に記録される:

```
# syslog.conf のサンプル
daemon.debug      /usr/adm/daemons
```

新しい仕組みのもとでもこの記述は全く同じ動作をもたらす。アスタリスク(*)、イコール記号(=)、エクスクラメーションマーク(!)、マイナス記号(-)の四つの新たな記述子が追加された相異点である。

*を指定すると、指示した facility に関するの全てのメッセージを一つに集めることができる。この動作は特定の priority レベルに対するデバッグに支障がでるかもしれない点に注意すること。このアスタリスク

記法はより直観的なものであることがわかるだろう。

=ワイルドカードは記録を指示された priority のもののみに限定する。たとえば特定のロギングについて、デバッグメッセージのみを集めることが可能となる。

下記の syslog.conf の記述例はすべての debug メッセージを/usr/adm/debug に記録する。

```
# syslog.conf のサンプル
*.=debug      /usr/adm/debug
```

priority の先頭に付く!は、上記の priority、記述子の解釈を反転する(訳注:すなわち、!info なら info 未満の priority を表す)。

以下の例では、priority が info であるものを除くすべての mail facility のメッセージが/usr/adm/mail ファイルに記録される。そして news.info(これは含む)から news.crit(こちらは含まれない)までのすべてのメッセージが/usr/adm/news ファイルに記録される。

```
# syslog.conf のサンプル
mail.*;mail.!=info     /usr/adm/mail
news.info;news.!crit     /usr/adm/news
```

除外する指示子はもっと直観的にも利用できる。上述の例はもっと簡単にできる。たとえば…

```
mail.none
```

や

```
mail.!*
```

や

```
mail.!debug
```

と記述すると mail facility によるすべてのメッセージが除外される。

-をファイル名に接頭すると書き込み時のファイルシステムバッファのフラッシュ動作を抑制することができる。

純粋な BSD 的挙動からは多少順応した結果なのかもしれないが、使う立場からすれば、BSD 的挙動よりもよりいくらかでも柔軟であることがわかるだろう。これらの変更は標準的な syslog.conf(5)ファイルにはなんら影響を及ぼしていない点に注意せよ。拡張機能を利用するには明示的に設定ファイルを調整する必要がある。

リモートロギングサポート

syslogd の機能にネットワークへのサポートを提供する変更点がいくつかある。ネットワークサポートとは、syslogd が稼働しているあるホストでのメッセージを別の syslogd が稼働しているホストへ転送し、そこで

実際にディスクのファイルに記録するようにできる、ということである。

この機能を有効にするにはコマンドラインで-r オプションを指定する。syslogd の既定の動作はネットワークを関知しない。

syslogd はローカルに生成されるメッセージについて UNIX ドメインソケットを監視する方法を用いる。この方法が syslogd に標準 C ライブラリに含まれる syslog との協調動作を可能にしている。同時に syslogd は他のホストから送信されるメッセージのために標準の syslog ポートを監視している。正しい動作のためには services(5)ファイル(普通/etc にある)に次のエントリが必要である:

```
syslog      514/udp
```

もしこのエントリがなければ、UDP ポートが開設できないために syslogd はリモートのメッセージを受信することも他へ送信することもできない。この場合、syslogd は即座に終了する代りにエラーメッセージを出力する。

メッセージを送信するべきホスト名に@を接頭して syslog.conf ファイルの通常のファイル名のかわりに記述することで、他のホストへメッセージを送信することができる。

たとえば「すべて」のメッセージをリモートのホストへ送信するには、syslog.conf に次のように記述する:

```
# メッセージをすべてリモートのホストへ
# 送信するための syslogd 設定ファイルの例
*.*     @hostname
```

すべてのカーネルメッセージをリモートのホストに送信するには (syslog.conf に)次のように記述する:

```
# カーネルメッセージをリモートのホストへ
# 送信する設定ファイルの例
kern.*     @hostname
```

起動時にネームサーバ(通常、syslogd よりも後から起動する)が応答しないためにリモートのホストネームが名前解決できなくても問題はない。syslogd はホスト名の問い合わせを 10 回繰り返し、そののちにエラーメッセージ出す。あるいはこの問題を回避するために、/etc/hosts ファイルに当のホスト名を記述しておくという方法もある。

普通の syslogd では、リモートホストが実は自分自身であった場合(または、より複雑な三角関係とか、そんなの)syslog-loop が発生する。たとえば作者のドメイン(Infodrom Oldenburg)でもたった一つのメッセージがわれわれのディスクをあふれかえさせるという事故が起きたことがある

これをさけるべく、リモートホストから(または自分自身)から発信されたメッセージはそれ以上転送されない。これでもまだ問題があるような状況があるのなら作者(Joey)まで連絡してほしい。

もしリモートのホストが syslogd が稼働しているホストと同じドメインに属しているのであれば、FQDN ではなくて単純にホスト名のみが記録される。

ローカルネットワークにおいて、重要な情報のすべてを一台のコンピュータに集めるための中央ロギングサーバを提供することができる。もしネットワークが複数のドメインからなっているような場合には、単純なホスト名ではなくて FQDN で記録されるが、それが嫌な場合は、-s でそのサーバで strip-domain 機能を使えばよい。これで syslogd はサーバが属するドメイン以外であっても剥ぎ取って単純にホスト名のみを記録する。

-l オプションを使用するとローカルのコンピュータとしてのホスト名を定義できる可能性が生じる。この場合でもやはり FQDN ではなくて単純なホスト名だけを記録する。

リモートホストへメッセージを転送したり、リモートホストからメッセージを受け取るために用いられる UDP ソケットは、それが必要な時にだけオープンされる。1.3-23 より前のリリースでは UDP ソケットは毎回オープンされていたが、読み込み用あるいは転送用という形ではなかった。

名前付きパイプ(FIFO)への出力

この版の syslogd は複数の名前付きパイプ(FIFO)へのログ出力も可能である。記録するメッセージを FIFO あるいは名前付きパイプで記録するには、ファイル名の前にパイプ記号(‘‘|’’)を付ける。これはデバッグに役に立つ。使用する FIFO は syslogd の起動に先立って mkfifo コマンドで作成しておかなければならない点に注意すること。

下記はカーネルのデバッグメッセージを FIFO で記録する為の設定例である。

```
# 名前付きパイプとしての/usr/adm/debug へ
# カーネルのデバッグメッセージを記録するための
# 基本的な設定例
kern.=debug      |/usr/adm/debug
```

インストール関連

この版の syslogd をインストールする場合において、重要な考察を要する点が多分一つだけある。この版の syslogd は syslog ファンクションによるメッセージのフォーマットが適切なものであることを前提としている。共有ライブラリにより提供される syslog ファンクションの動作内容は、libc.so.4.[2-4].n の範囲のなかだけでもどこかしら変更されている。なかでも明らかな変更は/dev/log ソケットへ出力される際にメッセージが NULterminate(¥0 で終端されること)されるようになったことである。よって、この版の syslogd はメッセージが¥0 で終端されていることに依存することとなった。

この問題は、概して、システム上で古いスタティックにリンクされたバイナリコードを動作させるときにあらわになる。古い syslog ファンクションを用いるバイナリコードは、メッセージ中の最初の文字が削除されたメッセージを記録し、空行を(複数)作ってしまう。この問題を修正するには、新しい版の共有ライブラリを用いてリンクしなおすしかない。

syslogd(8)と klogd(8)の両方とも init(8)を用いる起動でも rc.*の中で起動でもどちらでも可能である。もし、

init を用いる場合は-n オプションを必ず設定すること、さもなくばたくさんの syslog デーモンが起動してしまう。これは init(8)がプロセス ID に依存しているためである。

セキュリティ上の注意

syslogd デーモンは使用不能攻撃の抜け道として利用されてしまう潜在的な可能性を持っている。Jon Morrison(jmorrison@rflab.ee.ubc.ca)がこの可能性を警告した。ごろつきのプログラマ(達)が、syslog メッセージを syslogd デーモンに殺到させて、その結果、ログファイルで全ての利用可能なファイルシステムを埋め尽すことが簡単にできるようになっていた。インターネットドメインソケットを用いるロギングを有効にすることはローカルのコンピュータの上のプログラムや、他の誰かによる外部からの危険にシステムをさらすことになりうる。

コンピュータを守るための手段がいくつかある:

- 514/UDP ソケットにアクセスすることができるホストやネットワークを制限するファイアウォールをカーネルに実装する。
- ログの出力先をもしそれが破壊されてもコンピュータを損なうことはないような独立しているかまたはルート(/)ではないファイルシステムにする。
- ext2 ファイルシステムはそのうちの特定の割合を root だけが使用可能とする制限を設定することができる。注意この方法は syslogd を root ではないプロセスで実行する必要がある。さらに注意この方法は syslogd が 514/UDP ソケットと接続できなくなるので、リモートロギングが不可能となる。
- ローカルのコンピュータへの危険を制限するためにインターネットドメインソケットを無効にする。
- ステップ 4 を用いてもなお問題が残っていて、それが 3.5 フィート(だいたい 1 メートル)の吸出し棒(注)を持ったごろつきのプログラム/デーモンどころのさわぎではないようであれば、問題を起しているユーザとおしゃべりしてみるしかないね。

注:吸出し棒とは窶髏 3/4、7/8、1 インチの鋼鉄の棒で両端に吸い口が付いている。もともとは西ノースダコタの油田などで使われた油井から油を‘吸い出す’ポンプで使われているもののことである。それが転じて牧場で牛に餌をあたえるため、また時には反抗的だったり喧嘩腰だったりする牛を追うために使われている棒を差す。

デバッギング

-d オプションを用いてデバッグモードを有効にすると syslogd はとてつもなく饒舌になって、いまなにがおこっているかを標準出力に出力する。設定ファイルが再度読み込み込まれ解釈され直すときはいつでもテーブル化された内部データ構造が出力される。このテーブルは四つのフィールドから成っている:

- number

このフィールドは 0 から始まるシリアル番号である。この番号は内部データ構造上(すなわち配列)での位置をあらわす。もし番号がないときは、/etc/syslog.conf の対応する行にエラーがある。

- pattern

  このフィールドは巧妙で正確に内部構造を表現している。各列は facility(syslog(3)を参照)を表わす。以前使用されていたいくつかの facility もまだ残っているが、使用されているものだけがある。列の各フィールドは priority(syslog(3)を参照)をあらわす。

- action

  このフィールドには(facility/priority の)パターンに一致するメッセージを受信したときの action の詳細が記述される。全ての action については syslog.conf(5)マニュアルページを参照すること。

- arguments

  このフィールドは action の最後のフィールドによる付加的な引数を示す。ファイルロギングではログファイルとするファイル名;ユーザロギングでは(ログ出力を通知する)ユーザの一覧;リモートロギングではログを配信するコンピュータのホスト名;コンソールロギングでは使用されるコンソール;tty ロギングでは指定された tty;一斉通知の場合は付加引数はなし。

ファイル

- /etc/syslog.conf – syslogd の設定ファイル。正確な情報は syslog.conf(5)を見ること。
- /dev/log – ローカル syslog メッセージを読み出す UNIX ドメインソケット。
- /var/run/syslogd.pid – syslogd のプロセス ID を保持するファイル。

バグ

ある行にエラーがあると全てのルールを無視してしまう。

syslogd はその処理過程のどの時点においてもログファイルのファイルモードを変更しない。もしファイルが誰でも読み書きできるように作成されていても、である。もしこのことによる問題を回避したいのであれば作成時に管理者のみがあつかえるように変更する必要がある。

これは、smail 3.x の配布に含まれる savelog(8)によるログファイルのローテーションと連携して都合がよい。auth.*メッセージはパスワードが含まれていることがあり、それが誰にでも読めるということは重大なセキュリティホールになるという点を忘れないように。

関連項目

syslog.conf(5)、klogd(8)、logger(1)、syslog(2)、syslog(3)、services(5)、savelog(8)

6.2.4.2 カーネルログデーモン (klogd)

書式

```
klogd [ -c n ] [ -d ] [ -f fname ] [ -il ] [ -n ] [ -o ] [ -p ] [ -s ] [ -k fname ] [ -v ] [ -x ] [ -2 ]
```

説明

klogd は Linux のカーネルメッセージを捕え記録するシステムデーモンである。

オプション

- -c n

  コンソールに出力するログのレベルの既定値を n にする。

- -d

  デバッグモード。これは大量に stderr に出力する。

- -f file

  syslog の facility ではなくて指定した名前のファイルにメッセージを記録する。

- -i -I

  現在実行されている klogd デーモンにシグナルを送る。どちらのオプションもシンボル情報を(再)読み込みするように指示する。-i オプションはカーネルモジュールシンボルを再読み込みさせる。-I オプションは静的カーネルシンボルとカーネルモジュールシンボルの両方を再読み込みさせる。

- -n

  自動的なバックグラウンドへの移行を抑止する。これは klogd が init(8)により起動および制御される場合にのみ必要である。

- -o

  'ワンショット' モードで実行する。klogd はカーネルメッセージバッファに存在する全てのメッセージを読み出し記録する。一回の読み出しと記録ののちデーモンは終了する。

- -p

  パラノイアモード。klog がいつカーネルモジュールシンボルを読み込むかを指定する。このオプションを設定すると、カーネルメッセージストリームに"Oops"の文字列が流れる毎にカーネルモジュールシンボルの情報を読み込む。

- -s

  klogd はカーネルメッセージバッファとのインターフェイスにシステムコールの使用を強行する。

- -k file

  カーネルシンボル情報を指定した名前のファイルから取得する。

- -v

  バージョンを出力し、終了する。

- -x

  EIP 変換を抑制し、System.map を読み込まない。

- -2

  シンボルが展開された時、アドレスをシンボルに変換したものと生テキストの 2 回表示する。これにより、ksymoops のような外部プログラムが変換される前のデータを使って処理を行えるようになる。

概説

klogd の機能はよく他の版の syslogd に編入されてしまいがちであるが、それはあまり良い方法とは思われない。最近の Linux カーネルにおいては、情報源の特定、順位付け、カーネルアドレスの解決など多くのメッセージに関する問題を扱わなければならない。カーネルロギングを個別のプロセスとすることは、各種サービスの分割を明確なものにする。

Linux ではカーネルログ情報の情報源として二つの可能性がある。/proc ファイルシステムと syscall(sys_syslog)インターフェースであるが、突き詰めていけばこれらは同じ一つのものである。klogd は最もふさわしい情報としてどちらかを選択するように設計されている。最初に、実際にマウントされている/proc ファイルシステムを確認する。もしそこに/proc/kmsg ファイルがあれば、それをカーネルログ情報の情報源として利用する。もし proc ファイルシステムがマウントされていなければ、klogd はカーネルメッセージの取得にシステムコールを利用する。コマンドラインスイッチの(-s)は klogd にその情報源としてシステムコールの利用を強行させる。

もしカーネルメッセージが syslogd デーモンに振り向けられたとしても、version1.1 の klogd デーモンはその優先順位を適切に判定することが可能である。カーネルメッセージの優先順位付けは version 0.99pl13 あたりのカーネルで実装された。生のカーネルメッセージの形式は次の通り:

<[0-7]>カーネルからの出力

カーネルメッセージの優先順位は<>括弧に閉じられた一桁の数字に変換される。この数値はカーネルの include ファイル kernel.h で定義されている。カーネルからメッセージを受けると klogd デーモンはこの優先順位を読み取り、適切な syslog のメッセージレベルに割り付ける。(-f)によってファイルへの出力が指示されている場合には、カーネルメッセージに優先順位番号が残される。

klogd デーモンはカーネルメッセージの出力先をシステムコンソールへ変更することもできる。カーネルによって優先順位が付けられる結果として、メッセージはそれぞれデフォルトの既定のカーネルへのメッセージレベルが割り当てられている。手を加えていないカーネルのデフォルトのコンソールへのメッセージレベルは 7 に設定されている。7 よりも小さい(つまり高い)優先順位レベルを持つメッセージはコンソールに出力される。

レベル 7 の優先順位を持つメッセージは ‘debug’ メッセージとみなされ、コンソールには出力されない。特にマルチユーザ環境における、多くのシステム管理者は全てのカーネルメッセージを klogd により管理させ、ファイルか syslogd デーモンに渡したいと思うだろう。そうすれば、プリンタの用紙切れとかディスクの交換検出のような ‘わずらわしい’ メッセージのコンソールへの出力を避けることができる。

-c オプションが指定されると、klogd デーモンはコンソールに表示される全てのカーネルメッセージを抑制するシステムコールを実行する。以前のバージョンでは常にこのシステムコールが実行され、そのデフォルトは panic を除くすべてのカーネルメッセージであった。最近のバージョンでは少し違う扱いをしており、もはやこのオプション値を設定する必要はない。-c オプションの引数にはコンソールへ出力すべきメッセージの優先順位レベルを指定する。指示される数字「よりも小さい」優先順位値を持つメッセージがコンソールへ出力される、という点に注意すること。

たとえば、優先順位値が 3(KERN_ERR)かそれよりも重要なすべてのメッセージをコンソールに出力するためには、次のコマンドを実行する:

```
klogd -c 4
```

カーネルメッセージの(優先順位の)数値は、カーネルのソースコードがインストールされているのであれば、/usr/include/linux にある kernel.h ファイルで定義されている。これらの数値は/usr/include/sys サブディレクトリにある syslog.h ファイルでの優先順位値の定義に対応している。

klogd デーモンはカーネルメッセージを読み出す’ワンショット’モードも利用可能である。ワンショットモードはコマンドラインの-o オプションで指示される。その出力は syslogd デーモンに渡されるか-f スイッチが指定されていれば代りのファイルに書き出される。

たとえば、システムがブートした際のカーネルメッセージを全て読み出して、それを krnl.msg という名前のファイルに記録するには次のコマンドを実行する。

```
klogd -o -f ./krnl.msg
```

カーネルアドレスの解決

カーネルが内部エラー状態を検出すると、一般保護違反(General Protection Fault)が発生する。GPF 処理手続きの一部として、カーネルは違反が発生した時点でのプロセッサの状態を示すステータス報告を表示する。この表示にはプロセッサのレジスタの内容、カーネルスタックの内容、違反が発生した時にどの関数が実行されていたかのトレースが含まれる。

この情報は内部エラー状態が発生した原因を特定するために極めて重要である。カーネル開発者がこの情報を分析しようとすると、困難が生じる。なぜならカーネルは全て同じなわけではなく、変数の位置や関数のアドレスはカーネルごとに異なるからである。エラーの原因を診断するためには、カーネル開発者は特定のカーネルの、どの関数や変数位置がエラーに関係したかを知る必要がある。

カーネルコンパイル処理の一部として、コンパイルされたカーネルにおける重要な変数と関数のアドレスを記した一覧が作成される。この一覧はカーネルディレクトリソースツリーのトップに System.map という名前で作成される。この一覧を使って、カーネル開発者はエラー状態が発生した時にカーネルが何をしていたかを正確に知ることができる。

保護違反の表示から数値表現のアドレスを解決する処理は、手動かまたはカーネルソースに含まれる ksymoops プログラムを使って行なわれる。

利便性のために、klogd はカーネルの数値表現のアドレスを、それらのシンボル表現に変換しようとする。ただし実行時にカーネルのシンボルテーブルが必要である。もしシンボルの元のアドレスも必要な場合は、-2 を使うと数値アドレスも保存される。シンボルテーブルはコマンドラインの-k オプションを用いて指定する。シンボルファイルが明示されない場合は、次の順番でファイルを探す:

- /boot/System.map
- /System.map
- /usr/src/linux/System.map

カーネル 1.3.43 のシステムマップから、バージョン情報も提供されるようになっている。バージョン情報はシンボルテーブルのリストを解析検索する際に利用される。この機能は(カーネルの)安定版と先進版の両方で提供されているので(その判別に)役に立つ。

たとえば、安定版のカーネルはそのマップファイルを/boot/System.map に持っている。もし先進版のカーネルが/usr/src/linux の‘標準の’配置でコンパイルされているのであれば、システムマップは /usr/src/linux/System.map に存在する。klogd は先進版のもとで起動する時には /usr/src/linux/System.map マップファイルを優先して利用し、/boot/System.map マップファイルは無視する。

1.3.43 以降の最近のカーネルでは klogd がきちんと理解し、変換できるように重要なカーネルアドレスは適切に整列されている。それ以前のカーネルはカーネルのソースコードへのパッチが必要であり、そのパッチは sysklogd のソースコードと共に提供されている。

カーネル保護違反の分析処理は、静的カーネルに対しては非常にうまくいく。ローダブルカーネルモジュールで発生したエラーを診断しようとするとさらなる困難に出会うことになる。ローダブルカーネルモジュールはカーネルの機能の一部を自由にロードしたりアンロードしたりするのに用いられる。ローダブルモジュールはデバッグの観点から有用であり、カーネルが必要とするメモリの量を減らすのにも有用である。

ローダブルモジュールのエラー診断が困難なのは、カーネルモジュールが動的であるということによる。モジュールがロードされるとカーネルはモジュールを保持するためのメモリを確保し、モジュールがアンロードされるとこのメモリはカーネルに返される。動的にメモリが確保されるため、カーネルローダブルモジ

ュールの変数や関数のアドレスの詳細を記したマップファイルを作成することは不可能である。マップファイルなしではカーネルモジュールによる保護違反が発生した時にカーネル開発者が何が悪いのかを判断することは不可能である。

klogd はカーネルローダブルモジュールで発生した保護違反を診断する際に生じるこの問題を扱えるようになっている。プログラム開始時やシグナルを受け取った時に、klogd は全てのロードされているモジュールとそれらがロードされているメモリアドレスの一覧を問い合わせる。これらの外部シンボルのアドレスもこの問い合わせ処理の間に決定される。

保護違反が発生すると、静的シンボルテーブルからカーネルアドレスの解決を試みる。これに失敗した場合、現在ロードされているモジュールのシンボルを用いてアドレスの解決を試みる。これにより、最小限ではあるが、klogd は保護違反を起こしたローダブルモジュールがどれかを示すことができるようになる。もしモジュール開発者がモジュールからシンボル情報をエクスポートするようにしていれば、追加の情報も得られる。

カーネルモジュールのアドレスを適切かつ正確に解決するためには、カーネルモジュールの状態が変わる度にそれを klogd に知らせる必要がある。-i と-I オプションは現在起動しているデーモンにシンボル情報を再読み込みするように指示するために使われる。ほとんどの場合、適切にモジュールシンボルを解決させるために必要なのは-i オプションである。カーネルモジュールが追加または削除される度に、以下のコマンドを実行するべきである。

klogd –I

-p オプションもカーネルシンボル情報が最新であることを保証するために用いられる。このオプションは、保護違反が発生する度に klogd にモジュールシンボル情報を再読み込みするように指示する。プログラムを「パラノイア」モードで動かす前に注意してほしい。保護違反が発生した時のカーネルと実行環境の安定性は常に疑問である。モジュールシンボル情報を読み込むために klogd デーモンがシステムコールを実行する必要があるため、システムが不安定になって有用な情報が得られなくなる可能性がある。モジュールがロード・アンロードされた時に klogd(の情報)が更新されることを保証する方が遥かによい方法である。最新のシンボル情報をあらかじめ読み込んでおくことにより、保護違反が起きた時にそれを正しく解決する可能性が上昇する。

sysklogd のソースパッケージには modules-2.0.0 パッケージに対するパッチが含まれている。このパッチを適用すると、insmod、rmmod、modprobe を使ってカーネルにモジュールを追加・削除した時に自動的に klogd にシグナルを送るようになる。

シグナルの処理

klogd は以下の 8 種類のシグナルに反応する: SIGHUP、SIGINT、SIGKILL、SIGTERM、SIGTSTP、SIGUSR1、SIGUSR2、SIGCONT。このうち SIGINT、SIGKILL、SIGTERM、SIGHUP の各シグナルはデーモ

ンにカーネルログの生成源を閉じさせ、適切に終了させる。

SIGTSTP と SICONT の両シグナルはカーネルロギングの開始と終了のために利用される。SIGTSTP シグナルを受信するとデーモンはそのログの生成源を閉じ、アイドルループに突入する。その次に SIGCONT を受信するとデーモンは初期化を実行したのち、その入力源を再度選択し実行を再開する。SIGSTOP と SIGCONT の組合せは無停止でカーネルログの入力源を再選択させることができる。例えば、/proc ファイルシステムの利用を解除するには次の順番でコマンドを実行すればよい:

```
# kill -TSTP pid
# umount /proc
# kill -CONT pid
```

LOG_INFO 優先順位を持つシステムログがその停止/再開を記録する。

SIGUSR1 と SIGUSR2 はカーネルシンボル情報を(再)読み込みさせるために用いる。SIGUSR1 はカーネルモジュールシンボルを再読み込みさせる。SIGUSR2 は静的カーネルシンボルとカーネルモジュールシンボルの両方を再読み込みさせる。

System.map ファイルが適切な位置に置かれているなら、最も有効なシグナルは一般に SIGUSR1 である。このシグナルはカーネルモジュールが(再)読み込みされた時のために用意されている。カーネルモジュールの状態が変わった後にこのシグナルをデーモンに送れば、カーネルモジュールが占めているアドレス空間で保護違反が起きた時に適切にシンボルを解決できることが保証される。

ファイル

- /proc/kmsg - klogd の記録するカーネルメッセージ源の一つ
- /var/run/klogd.pid - klogd のプロセス id が記録されているファイル
- /boot/System.map、/System.map - カーネルシステムマップのデフォルト位置
- /usr/src/linux/System.map - カーネルシステムマップのデフォルト位置

6.2.4.3 システムログにエントリを作成する (logger)

書式

```
logger [-isd] [-f file] [-p pri] [-t tag] [-u socket] [message ...]
```

説明

logger はシステムログにエントリを作成する。logger は syslog(3)システムログモジュールのシェルコマンドインターフェースを提供する。

オプション

- -I

  各行に logger プロセスのプロセス ID を記録する。

- -s

  システムログに記録したメッセージを標準エラー出力にも出力する。

- -f file

  指定したファイルの内容を記録する。

- -p pri

  メッセージを指定した優先度(priority)で登録する。優先度は数値もしくは ‘ ‘機能分類.重要度’ ’ の組で指定する。例えば、‘ ‘-plocal3.info’ ’ は、重要度 informational 機能分類 local3 としてメッセージを記録する。デフォルトは ‘ ‘user.notice’ ’ である。

- -t tag

  ログを出力する各行に、指定した tag を共に記録する。

- -u sock

  組み込みの syslog ルーチンの代わりに sock で指定されたソケットに出力する。

- -d

  このソケットへのストリーム接続ではなく、データグラムを使う。

- --

  引数の終わり。message をハイフン(-)で始められるようにする。この機能はオリジナルの BSD logger コマンドにはない。GNU 拡張である。

- message

  log ファイルに書き込むメッセージ。これが指定されず、かつ-f オプションも指定されなかった場合は、標準入力からの入力が記録される。

指定できる機能分類名は以下の通り:auth、authpriv(機密に関わる種類のセキュリティ情報)、cron、daemon、ftp、kern、lpr、mail、news、security(auth の同義語。使わない方が良い)、syslog、user、uucp と、local0～local7。

指定できる重要度は以下の通り:alert、crit、debug、emerg、err、error(err の同義語。使わない方が良い)、info、notice、panic(emerg の同義語。使わない方が良い)、warning、warn(warning の同義語。使わない方が良い)。これらの重要度の優先順位と意図する目的については、syslog(3)を参照すること。

返り値

logger ユーティリティは成功した場合 0 を返し、エラーの場合は 0 より大きい値を返す。

例

```
logger System rebooted
logger -p local0.notice -t HOSTIDM -f /dev/idmc
```

関連項目

syslog(3)、syslogd(8)

6.2.4.4 予定されたコマンドを実行するデーモン (cron)

書式

cron

説明

cron は/etc/rc または/etc/rc.local から起動されるべきである。すぐに(シェルに)戻るので、'&'を付けて起動する必要はない。

cron は/etc/passwd にあるアカウントをファイル名に持つ crontab ファイルを/var/cron/tabs から探し、見つけた crontab ファイルをメモリに読み込む。また cron は/etc/crontab も見る(このファイルのフォーマットは少々異なっている:crontab(5)を参照)。cron は 1 分ごとに起きて、読み込まれた crontab ファイルを評価し、それぞれのコマンドを今起動すべきかどうかチェックする。コマンドを実行すると、全ての出力を crontab ファイルの所有者にメールする(または MAILTO 環境変数が crontab ファイルにあれば、そこで指定されたユーザーに送る)。

さらに cron は 1 分ごとにスプールディレクトリ(または/etc/crontab ファイル)の最終修正時刻(modtime)をチェックし、もし変更されていれば、すべての crontab ファイルの最終修正時刻をチェックし、変更された crontab ファイルを読み直す。よって crontab ファイルを修正するたびに cron を再起動する必要はない。crontab(1)コマンドは、crontab ファイルが変更されたかどうかにかかわらず、スプールディレクトリの最終修正時刻を更新することに注意せよ。

関連項目

crontab(1)、crontab(5)

6.2.4.5 各ユーザーのための crontab ファイルを管理する（crontab）

書式

```
crontab [ -u user ] file
crontab [ -u user ] { -l | -r | -e }
```

説明

crontab は、VixieCron パッケージの cron(8)デーモンの運用に使われるテーブルをインストール・アンインストール・表示するためのプログラムである。ユーザーはそれぞれ自分用の crontab を保有できる。これらは/var 以下に置かれるが、直接には編集できないようにしてある。

allow ファイルが存在する場合、ユーザーがこのコマンドを使用するには、そのファイル中に（そのユーザーが）リストアップされている必要がある。allow ファイルは存在せずに deny ファイルが存在する場合、ユーザーがこのコマンドを使用するには、deny ファイル中にリストアップされていない必要がある。いずれのファイルも存在しない場合、スーパーユーザーのみがこのコマンドを使えるか、あるいはすべてのユーザーがこのコマンドを使えることになり、そのいずれであるかはサイトに依存した設定パラメータによって決まる。

-u オプションでは、対象となる crontab の所有者名を指定する。このオプションが与えられていない場合、crontab は"あなたの"crontab、すなわちコマンドを実行している人の crontab を調べる。なお、su(8)を使っているとcrontabを混同しかねないため、su(8)内部での実行中は、安全のため常に-uオプションを使うべきである。

このコマンドの１つ目の書式は、新しいcrontabを（何らかの名前の付けられた）ファイル、もしくは標準入力（疑似ファイル名 ‘ ‘-’ ’ が与えられた場合）からインストールするために使われる。

-l オプションは、現在の crontab を標準出力へ表示させる。

-r オプションは、現在の crontab を削除する。

-e オプションは、環境変数 VISUAL もしくは EDITOR で指定されているエディターを使って、現在の crontab を編集するのに使われる。編集終了後、変更された crontab は自動的にインストールされる。

関連項目

crontab(5)、cron(8)

ファイル

- /var/cron/allow

- /var/cron/deny

準拠

crontab コマンドは、IEEE Std1003.2-1992('POSIX')に準拠している。この新しいコマンドのシンタックスは、Vixie Cron の前のバージョンと異なっている(古典的な SVR3 シンタックスとも異なる)。

返り値

正しくないコマンドラインでこのコマンドを実行すると、多少の情報を含む利用法のメッセージが表示される。

6.2.4.6 ログのローテーションを行う (logrotate)

書式

```
logrotate [-dv] [-f|--force] [-s|--state file] config_file+
```

説明

logrotate は多数のログファイルを生成するシステムの管理を容易にする。それはログのローテーション、圧縮、削除、ログファイルのメール送信を自動で行う。それぞれのログファイルは日ごと、週ごと、月ごと、またはサイズが大きくなりすぎる前に実行される。

通常、logrotate は日ごとの cron ジョブによって実行される。ログローテーションの基準がログサイズで、1 日に複数回 logrotate が動作するように設定されているか、-f または--force オプションを使用しない限り、logrotate は 1 日に複数回ログローテーションを実行しない。

複数の設定ファイルはコマンドラインによって与えられる後から読み込まれた設定ファイルは、先に読み込まれた設定を上書きするため、指定する設定ファイルの順序は重要である。通常、他のいくつかの設定ファイルを include した 1 つの設定ファイルを利用する。include パラメータの利用方法はパラメータの項を参照のことコマンドラインにディレクトリが指定さえ r た場合、そのディレクトリ下のファイルが全て読み込まれる。

コマンドラインの引数が何も与えられなかった場合、logrotate は著作権情報と短い利用方法を表示する。ログローテーション中にエラーが発生した場合、logrotate は 0 以外の終了ステータスで終了する。

オプション

- -d

  デバッグモードを有効にし、-v オプションを付加するデバッグモードでは、ログファイルや

logrotate のステートファイルは作成されない。

- -f、--force

  強制的にローテーションを実行する。これは logrotate に新しいエントリを追加した場合や、古いログファイルを手動で削除した場合に、新しいファイルを作成して正常にログを記録させるために役立ちます。

- -m、--mail <command>

  ログをメールする場合に、どのコマンドを使用するかを設定する。このコマンドは 2 つの引数を受け付ける: 1) メッセージのサブジェクト、2) 受信者。コマンドは標準入力からメッセージを読み、受信者にメールを送信する。デフォルトのメールコマンドは/bin/mail -s である。

- -s、--state <statefile>

  ステートファイルの場所を指定する。異なるユーザーで複数の logrotate を動作させる場合に有用である。デフォルトのステートファイルは/var/lib/logrotate/status である。

- --usage

  短い usage メッセージを表示する。

設定ファイル

logrotate はコマンドラインから指定された設定ファイルから、対象となるログファイルの情報を読み込む。それぞれの設定ファイルはグローバルオプションとローテーションを行うログファイルの設定することが可能である(ローカル設定はグローバル設定を上書きし、また、より後に読み込まれた設定が優先される)。簡潔な設定ファイルを以下に示す。

```
# sample logrotate configuration file
compress

/var/log/messages {
 rotate 5
 weekly
 postrotate
                                    /sbin/killall -HUP syslogd
 endscript
}

"/var/log/httpd/access.log" /var/log/httpd/error.log {
 rotate 5
 mail www@my.org
 size=100k
 sharedscripts
 postrotate
                                    /sbin/killall -HUP httpd
 endscript
}

/var/log/news/* {
 monthly
```

```
 rotate 2
 olddir /var/log/news/old
 missingok
 postrotate
                                        kill -HUP `cat /var/run/inn.pid`
 endscript
 nocompress
}
```

最初の数行はグローバルオプションである。例では、ログファイルはローテーション後に圧縮される設定である。先頭にスペースがなく、#で始まる行はコメントとして扱われる。

次のセクションはログファイル/var/log/messages の設定パラメータである。週ごとのローテーションを 5 回行った後、最も古いものは削除される。ログファイルがローテーションされた後、古いログが圧縮される前に、/sbin/killall -HUP syslogd コマンドが実行される。

次のセクションは/var/log/httpd/access.logと/var/log/httpd/error.logの設定パラメータである。ログファイルが 100K よりも大きくなった時にローテーションされ、5 回のローテーションが行われた後、古いログは削除されるのではなく非圧縮で www@my.org にメールされる。sharedscripts は、複数指定したログファイルに対して postrotate または prerotate で記述されたコマンドを実行することを意味する。このセクションの先頭にあるファイル名の前後のダブルクォーテーションは、名前にスペースを含む場合に必要である。通常のシェルクォーティング規則が適用され、「'」「"」「¥」の文字が利用できる。

最後のセクションはログファイル/va/log/news 以下のファイル全ての設定パラメータである。各ファイルは月ごとにローテーションされ、圧縮は行われない。1 つ以上のファイルでエラーが発生した場合、ログローテーションは行われない。

そのためワイルドカードを使う場合は注意すること。*を設定した場合、以前にローテーションしたものを含め、全てのファイルがローテーションされる。olddir オプションを使用するか、*.log のように、より詳細なワイルドカード指定をすることによりこの問題は回避できる。

パラメータ

- compress

  古いファイルを gzip(デフォルト)で圧縮する。関連項目: nocompress。
- compresscmd

  ログファイルの圧縮に使用するコマンドを指定する。デフォルトは gzip。関連項目: compress。
- uncompresscmd

  圧縮されたログファイルの展開に使用するコマンドを指定する。デフォルトは gunzip。
- compressext

  ログの圧縮が有効になっている場合に、圧縮後に利用する拡張子を指定する。デフォルトでは圧縮に使用するコマンドに従う。

- compressoptions

  ログの圧縮が有効になっている場合に、圧縮に使用するコマンドに渡すオプションを指定する。デフォルトでは gzip に-9 オプションを使用する。

- copy

  ログファイルのコピーを作成し、オリジナルは変更しない。このオプションは、ログファイルのスナップショットを作成する場合や、その他のユーティリティーでログファイルを編集する必要がある場合などに有用である。このオプションを利用した場合、create オプションは無視される。

- copytruncate

  コピーを作成した後、元のログファイルの先頭部分を削除し、ファイルサイズを小さくする。このオプションは、ログファイルを閉じることができないプログラムを利用している場合に有用である。ファイルのコピーと先頭部分の削除の間には、非常に短い間隔があるため、ログデータが失われる場合があることに注意が必要である。このオプションを利用した場合、create オプションは無視される。

- create mode owner group

  新しいログファイルは、ローテーションが行われた直後(postscript が動作する前)に、同じ名称で作成される。mode は 8 進法(chmod と同様)で作成されるログファイルのモードを指定する。owner はユーザー名、group はグループ名をそれぞれ指定する。それぞれの属性は省略することが可能であり、省略された場合は元のログファイルと同じ属性が利用される。このオプションは nocreate オプションを使用することで無効にできる。

- daily

  ログファイルは毎日ローテーションされる。

- delaycompress

  ログファイルの圧縮を 1 サイクル遅らせて実行する。このオプションは conpress オプションとともに使用した場合のみ有効である。ログファイルを閉じることができず、しばらくの間、元のログファイルに書きつづけるようなプラグラムを利用している場合に有用である。

- extension ext

  ローテーション後に拡張し ext を付加する。圧縮が有効な場合、圧縮の拡張子(デフォルトは.gz)は ext の後に付加される。

- ifempty

  ログファイルが空であってもローテーションを行う。このオプションはデフォルトで設定されており、notifempty オプションで上書き可能である。

- include file_or_directory

  指定したファイルやディレクトリから設定ファイルを読み込む。ディレクトリが指定された場合、その下にあるファイルがアルファベット順で処理される。通常のファイルではない場合(ディレクトリや名前つきパイプなど)や tabooext で設定された拡張子を持つファイルは無視される。include オプションは必ずしも設定する必要はない。

- mail address

  ログがローテーションによって削除されるときに、ログファイルをメールで送信する。nomail オプションを使用することで、特定のログを送信しないように設定することができる。

- mailfirst

  メールコマンドを使った際に、ログファイルの削除のときではなくローテーションを行ったときにメールを送信する。

- maillast

  メールコマンドを使った際に、ログファイルのローテーションのときではなく削除されるときにメールを送信する(デフォルト)。

- missingok

  ログファイルが存在しなければ、エラーを出さずに次の処理を実行する。関連項目: nomissingok。

- monthly

  ログファイルを月ごとにローテーションする(通常は月の最初の日)。

- nocompress

  古いログファイルを gzip で圧縮しない。関連項目: compress。

- nocopy

  オリジナルのログファイルをコピーせず、そのままの状態にしておく。(copy オプションを上書きする)。

- nocopytruncate

  コピーを作成したあと、オリジナルのログファイルの先頭を削除しない(copytruncate オプションを上書きする)。

- nocreate

  新しいログファイルを作成しない(create オプションを上書きする)。

- nodelaycompress

  ログファイルの圧縮を 1 サイクル遅らせて実行しない(delaycompress オプションを上書きする)。

- nomail

  古いログファイルをメールで送信しない。

- nomissingok

  ログファイルが存在しなければ、エラーを出力する。デフォルトの状態で設定されている。

- noolddir

  同じディレクトリでログをローテーションする(override オプションを上書きする)。

- nosharedscripts

  ローテーションの際に prerotate と postrotate スクリプトを実行する(デフォルト。sharedscripts オプションを上書きする)。

- notifempty

  ログファイルが空であればローテーションを行わない(ifempty オプションを上書きする)。

- olddir directory

  ローテーションの際にログファイルをディレクトリに移動する。ディレクトリはログファイルと同じ物理デバイスに存在する必要があり、絶対パス名で指定する必要がある。このオプションを指定した場合、古いログは全てディレクトリに保存される。このオプションは noolddir オプションを上書きする。

- postrotate/endscript

  ログファイルがローテーションされた後に、postroteta と endscript の間の行が実行される。これらのオプションはログファイル設定の中でのみ利用することができる。関連項目: prerotate。

- prerotate/endscript

  ログファイルがローテーションされる前に、prerotate と endscript の間の行が実行される。これらのオプションはログファイル設定の中でのみ利用することができる。関連項目: postrotate。

- firstaction/endscript

  ローテーションされるファイルが 1 つ以上存在する場合に、prerotate スクリプトが実行される前に、ワイルドカードパターンにマッチした全てのログファイルに対して、1 度だけ firstaction と endscript の間の行が実行される。これらのオプションはログファイル設定の中でのみ利用することができる。関連項目: lastaction。

- lastaction/endscript

  ローテーションされるファイルが 1 つ以上存在する場合に、postrotate スクリプトが実行された後に、ワイルドカードパターンにマッチした全てのログファイルに対して、1 度だけ lastaction と endscript の間の行が実行される。これらのオプションはログファイル設定の中でのみ利用することができる。関連項目: lastaction。

- rotate count

  ログファイルが削除またはメール送信される前にローテーションされる数を count に指定する。もし count が 0 であれば、古いログファイルはローテーションは削除される。

- size size

  ログファイルのサイズが size バイトよりも大きくなった時にローテーションを行う。size の後に M が指定された場合はメガバイト、k が指定された場合はキロバイトとして扱う。

- sharedscripts

  通常、prescript と postscript のスクリプトは、それぞれのローテーションされたログに対して実行される。これは 1 つのスクリプトが複数のログ(ワイルドカードにマッチした場合など)に対して、複数回実行されることを意味する。sharedscripts が指定された場合、スクリプトは一度だけ実行され、複数のログがワイルドカードパターンにマッチした場合でも問題にならない。しかし、パターンにマッチするログが存在しない場合は、スクリプトは実行されない。このオプションは

nosharedscripts オプションを上書きする。

- start count

  ローテーションに使用する基本となる数字を指定する。0 を指定した場合、オリジナルのログファイル名に拡張子.0 を付けてローテーションされ、9 を指定した場合、0 繰・は使用せず、拡張子.9 を付けてローテーションされる。ログファイルは count パラメータで設定した数だけローテーションされる。

- tabooext [+] list

  現在の例外拡張子リストを変更する(例外拡張子については inclide オプションを参照)。+を付加した場合、現在のリストに追加され、付加しなければリスト全体が変更される。デフォルトでは.rpmorig、.rpmsave、.v、.swp、.rpmnew、~が含まれる。

- weekly

  現在の週日が前回ローテーションを行った週日と異なるなるか、前回のローテーションから 1 週間経過している場合にログをローテーションする。これは通常、週の始めの日にローテーションを行うことと同じであるが、logrotate が毎日実行される場合に有用である。

ファイル

- /var/lib/logrotate/status – デフォルトの state ファイル
- /etc/logrotate.conf – 設定ファイル

関連項目

gzip(1)

6.2.4.7 システムの日付と時刻を表示・設定する (date)

書式

date [-uR] [-d datestr] [-f datefile] [-r file] [-s datestr] [-I [timespec]] [--date=datestr] [--file=datefile] [--iso-8601[=timespec]] [--reference=file] [--set=datestr] [--rfc-822] [--universal] [--utc] [+format] [MMDDhhmm[[CC]YY][.ss]]

date [--help] [--version]

説明

date は引数を指定しないと、現在の時刻と日付を表示する(表示形式は '%a%b %e %H:%M:%S %Z %Y' となる。以下を参照のこと)。

引数が指定され、かつその先頭が '+' で始まっておらず、かつ実行者が適切な権限を持っていれば、date はシステムの時計を引数で指定された時刻・日付に設定する。--date および--set の両オプションは、このような引数と一緒に用いることはできない。--universal オプションをこのような引数とともに用いると、指定した時刻・日付が(地方時ではなく)協定世界時のものになる。引数には数字だけを用いることができ、それぞれ以下の意味を持つ:

- MM 月
- DD 日(月内通算)
- hh 時
- mm 分
- CC 年の最初の 2 桁(省略可)
- YY 年の最後の 2 桁(省略可)
- ss 秒 (省略可)

指定された引数が '+' で始まる場合には、date は現在の時刻と日付を表示する(あるいは--date オプションで指定された時刻と日付を指定する)。表示形式はこの引数によって制御され、引数の書式は strftime(3)関数にわたす文字列の書式と同じである。`%' で始まるフィールドを除き、format 文字列中の文字はそのまま変更されずに表示される。

時刻フィールド

- %H 時(00..23)
- %I 時(01..12)
- %k 時(0..23)
- %l 時(1..12)
- %M 分(00..59)
- %p AM あるいは PM のロケール
- %r 時刻、12 時間(hh:mm:ss[AP]M)
- %s 1970-01-01 00:00:00 UTC からの秒数(標準外の拡張)
- %S 秒(00..60)
- %T 時刻、24 時間(hh:mm:ss)
- %X ロケールによる時刻の表現(%H:%M:%S)
- %Z タイムゾーン(例 EDT)、あるいはタイムゾーンが決定できないならば無し

日付フィールド

- %a ロケールの省略形の曜日名(Sun..Sat)
- %A ロケールの完全表記の曜日名、可変長(Sunday..Saturday)
- %b ロケールの省略形の月名(Jan..Dec)

- %B ロケールの完全表記の月名、可変長(January..December)
- %c ロケールの日付と時刻(Sat Nov 04 12:02:33 EST 1989)
- %d 月内通算日数(01..31)
- %D 日付(mm/dd/yy)
- %h %b と同じ
- %j 年内通算日数(001..366)
- %m 月(01..12)
- %U 日曜日を週の最初の日とした年内通算週(00..53)
- %w 週のうちの曜日(0..6)(0 が日曜日)
- %W 月曜日を週の最初の日とした年内通算週(00..53)
- %x ロケールの日付表現(mm/dd/yy)
- %y 年の最後の 2 つの数字(00..99)
- %Y 年(1970...)

文字フィールド

- %% 文字%
- %n 改行
- %t 水平タブ

文字埋め(padding)

デフ 3 ォルトでは、date は数値のフィールドを 0 で埋める。したがって、例えば数値表示の月は常に 2 桁で出力される。GNU は dat の機能を拡張しており、以下の非標準の数値修正子を '%' とフィールドの間に置くことができる:

- - (ハイフン) フィールドを埋めない。出力が人に読まれる場合には便利である。
- _ (アンダースコア) フィールドをスペースで埋める。出力に決まった数の文字が必要だが、0 を使いたくない場合に便利である。

オプション

- -d datestr、--date datestr

  現在の時刻・日付の代わりに、datestr で指定された時刻・日付を表示する。datestr は普通のフォーマットならだいたいなんでも使うことができる。月名、タイムゾーン、'am' や 'pm' なども用いてよい。
- -f datefile、--file=datefile

  -d とともに指定された datefile を 1 行ずつパースして、結果の時刻と日付を表示する。datefile が '-' の場合は標準入力が用いられる。これは、たくさんの日付を処理しなければならない場

合に便利である。date の実行ファイルを何回も起動するオーバーヘッドは無視できないからである。

- -r file、--reference=file

  表示する時刻と日付を file の最終修正時刻にする。

- -s datestr、--set datestr

  時刻と日付を datestr に設定する。上述の-d を見よ。成功すると 0 を返し、失敗すると 0 以外を返す。

- -u、--universal、--utc

  タイムゾーンが地方時(壁時計の時刻)ではなくUTC0(協定世界時、これはグリニッジ平均時もしくは GMT として知られている)に設定されたものとする。

- -I [timespec]、--iso-8601[=timespec]

  日付を ISO8601 で指定されている書式 '%Y-%m-%d' で、時刻を timespec で指定されている形式で表示する(後者のデフォルトは auto)。時刻部分の表示には 'T' が前置され、'%z' (--utc が指定されている場合には '%Z')が後置される。

  auto　時刻を表示しない。

  hours　その日の時刻を表示する。

  minutes 時・分を表示する。

  seconds 時・分・秒を表示する。

- -R、--rfc-822

  時刻と日付を RFC-822 で指定された書式である '%a %_d %b %Y%H:%M:%S %z' で表示する。--utc が同時に指定されると、'%z' の代わりに 'GMT' を用いる。日付と月の名前は 'C' ロケールに基づいて表示される。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

例

一昨日の日付を表示するには:

```
date --date '2 days ago'
```

3ヶ月と1日後の日付を表示するには:

```
date --date '3 months 1 day'
```

今年のクリスマスが年の初めから何日目かを表示するには:

```
date --date '25 Dec' +%j
```

完全な月名と日付からなる書式で今日を表示するには:

```
date ´+%B %d´
```

しかしこの結果はお望みのものではないかもしれない。なぜなら月の最初の 9 日を表示させるとき、`%d´は 2 桁のフィールドの最初を 0 で埋めるからである。例えば`date -d 1-may ´+%B %d´´の結果は`May 01´となる。

同じ日を、1 桁の日付の前に 0 を置かないように表示するには、標準にはない`-´修正子を用いて 0 埋めを行わないようにすればよい。

```
date -d 1-may ´+%B %-d´
```

注意

プログラムのバグについては bug-sh-utils@gnu.org に報告してください。ページの更新は Ragnar Hojland Espinosa <ragnar@ragnar-hojland.com>が行っています。

6.2.4.8 faillog を調べ、login 失敗の制限を設定する (faillog)

書式

faillog [-u login-name] [-a] [-t days] [-m max] [-pr]

説明

faillog はログインの失敗を記録するログファイル/var/log/faillog の内容を整形し、失敗の回数及びその制限の保守を行なう。faillog に与える引数の順序には意味がある。各引数は与えられた順に直ちに処理される。

-p フラッグはログイン失敗のエントリを UID 順に表示させる。-u login-name と入力した際は、login-name のログイン失敗記録のみを表示する。-t days と入力すると、最近 days 日以内のログイン失敗の記録を表示する。-t フラッグを用いた際は-u フラッグの働きが抑制される。-a フラッグを用いると全ユーザに対する記録が表示される。このフラッグを-p フラッグとともに用いた場合は、これまでにログインに失敗した事のある全てのユーザが選び出される。-r フラッグと用いても意味が無い。

-r フラッグはログイン失敗の回数をリセットする為のものである。このオプションを用いるには/var/log/faillog への書き込み権が必要である。-u login-name と入力した際は、login-name のログイン失敗回数のみをリセットする。

-m フラッグは、アカウントが使用不能になる最大のログイン失敗回数を設定する為に用いる。このオプションを用いるには/var/log/faillog への書き込み権が必要である。-m max と入力した場合、ログイン失

敗が max 回に達した後は全てのアカウントが使用不能になる。-u login-name とともに用いればこの機能を login-name にのみ作用させる事が出来る。max の値を 0 にするとログインの失敗回数には制限が無くなる。アタックにより root がログイン不能となりシステム管理が出来なくなってしまう事を防ぐ為に、root に対しては最大ログイン失敗回数はいつでも必ず 0 としなくてはならない。

オプションはどのように組み合わせても良い。-p、-r 及び-m の各オプションは-u 又は-t のいずれと組み合わせても直ちに処理される。

警告

faillog は最後に失敗して以来ログインに成功していないユーザのみを表示する。ログインに失敗した後に正しくログイン出来たユーザを表示させる為には、-u フラッグを用いてそのユーザを明示的に指定するか、-a フラッグを用いて全ユーザを表示させなくてはならない。

システムによっては、/var/log の代わりに/var/adm 又は/usr/adm を用いていることもある。

ファイル

- /var/log/faillog - ログイン失敗を記録するファイル

関連項目

login(1)、faillog(5)

6.2.4.9 ハードウェア・クロック(RTC)の読み取りと設定を行う (hwclock)

書式

```
hwclock -r or hwclock --show
hwclock -w or hwclock --systohc
hwclock -s or hwclock --hctosys
hwclock -a or hwclock --adjust
wclock -v or hwclock --version
hwclock --set --date=newdate
hwclock --getepoch
hwclock --setepoch --epoch=year
```

その他のオプション:

```
[-u|--utc]  --localtime       --noadjfile --directisa --test [-D|--debug]
```

DEC Alpha 用オプション:

```
[-A|--arc] [-J|--jensen] [-S|--srm] [-F|--funky-toy]
```

すべてのオプションは他と区別がつく範囲において短縮することができる。また-h はヘルプメッセージを表示する。

説明

hwclock はハードウェア・クロックにアクセスするためのツールである。現在の時刻の表示、指定した時刻へのハードウェア・クロックの設定、ハードウェア・クロックをシステム時刻に合わせる(およびその逆)、といった機能を持つ。

hwclock を定期的に実行し、ハードウェア・クロックの時間を増減して、時計の規則的なずれ(systematic drift)を補償することもできる(systematic drift とは、クロックが放っておかれたとき、経過時間に比例して時刻がずれる現象のこと)。

オプション

以下のオプションは hwclock にどの機能を実行するかを伝えるもので、必ず一つだけを指定する。

- --show

  ハードウェア・クロックを読んで時刻を標準出力に表示する。ここで表示される時刻は常にローカル・タイムである。ハードウェア・クロックを協定世界時にしていても表示はローカル・タイムである。--utc オプションの部分を参照すること。
- --set

  ハードウェア・クロックを--date オプションによって指定した時刻に設定する。
- --hctosys

  システム・クロックをハードウェア・クロックに合わせる。同時にカーネルが持つタイムゾーンの値もローカルのタイムゾーンにセットする。このとき TZ 環境変数や/usr/share/zoneinfo の内容を tzset(3)と同じように解釈して参照する。カーネルのタイムゾーンの obsolete なフィールドである tz_dsttime は DST_NONE に設定される。(このフィールドがかつて意味していた内容に関しては settimeofday(2)を参照のこと。) このオプションはシステムの起動スクリプトの一部で用いるとよい。
- --systohc

  ハードウェア・クロックを現在のシステム・クロックに合わせる。
- --adjust

  最後にハードウェア・クロックを合わせた時点からの経過時間に対して生じる、時計の規則的なずれを補償するために、一定の時間をハードウェア・クロックの時刻から増減する。詳細は以下の議論を参照のこと。
- --getepoch

  標準出力に、カーネルが保持しているハードウェア・クロックの紀元年(epoch value)を表示す

る。これは西暦の何年が、ハードウェア・クロックの 0 年として参照されるかを示す数値である。例えば、ハードウェアクロックの年カウンタに1952年以降の経過年数を用いている場合には、カーネルでのハードウェア・クロック紀元年は 1952 でなければならない。この紀元年の値は、hwclock がハードウェア・クロックを読み書きするとき常に用いられる。

- --setepoch

  カーネルのハードウェア・クロック紀元年の値を--epoch オプションで指定した値に設定する。詳細は--getepoch オプションの説明を見よ。

- --version

  hwclock のバージョンを標準出力に表示する。

- --date=date_string

  --set オプションを指定した場合は、このオプションも指定しなければならない。--set オプションが指定されていなければ、このオプションは無視される。ハードウェア・クロックを合わせる時刻を指定する。このオプションに与える値は date(1)プログラムの引数と同じである。例えば以下のようにする。

  hwclock --set --date="9/22/96 16:45:05"

  引数はローカルタイムで与える。ハードウェア・クロックを協定世界時にしている場合でも、である。--utc オプションの部分を見よ。--setepoch オプションを指定した場合は次のオプションも必要である。

- --epoch=year

  ハードウェア・クロックの紀元年を指定する。すなわち西暦年のいつが、ハードウェア・クロックの年カウンタの 0 に対応するかを指定する。このオプションは、--setepoch オプションとともに使った場合、カーネルの概念であるハードウェア・クロックの紀元年を設定する。--setepoch オプションとともに使わない場合は、直接 ISA アクセスに用いられる紀元年を指定する。例えば、Digital Unix マシンでは以下のようにする。

  hwclock --setepoch --epoch=1952

次のオプションはほとんどの機能と同時に用いることができる。

- --utc、--localtime

  ハードウェア・クロックを協定世界時(Universal Coordinated Time: UTC)とローカルタイムのどちらにするか(しているか)を指定する。UTC にするかローカルタイムにするかはユーザの選択しだいだが、時計の内部にはどちらを選択したかを記録する場所はない。したがって、ユーザーはこのオプションで自分の選択を hwclock に伝えなければならない。

  これらの指定を間違ったほうにしたり(あるいはデフォルトを勘違いして両方とも指定しなかったり)すると、ハードウェア・クロックの設定やクロックへの問い合わせの結果はめちゃめちゃになってしまうだろう。

  --utc も--localtime も指定しなかった場合のデフォルトは、最後に hwclock を使って時計を合

わせたとき(つまり--set、--systohc、--adjust オプションを指定しての実行が成功したとき)に指定していた方になる。このときの選択は adjtime ファイルに記録されている。adjtime ファイルがなかったときのデフォルトはローカルタイムになる。

- --noadjfile

  /etc/adjtime によって提供される機能を無効にする。このオプションを使うと、hwclock は /etc/adjtime の読み込みも書き込みもしない。このオプションを使うときは、--utc または --localtime を指定しなければならない。

- --directisa

  このオプションは、ISA マシンまたは(hwclock から充分 ISA マシンに見える程度 ISA の仕様を実装した)Alpha マシンでのみ意味を持つ。他のマシンでは効果がない。このオプションは hwclock に指令して、ハードウェア・クロックへのアクセスに直接 I/O 命令を用いるようにさせる。このオプションを指定しないと、hwclock は/dev/rtc デバイスを用いようとする(/dev/rtc が rtc デバイスドライバで駆動されていることを仮定する)。デバイスを読み込みオープンできない場合は、いずれにせよ直接 I/O 命令を用いる。rtc デバイスドライバは Linux リリース 2 から現れた。

- --badyear

  ハードウェア・クロックが、1994-1999 年の外側の年を保持できないことを示す。ある種の BIOS には問題があり(4/26/94 から 5/31/95 の間に生産されたほとんどの Award BIOS がそうである)、1999 年以降の年を扱うことができないのである。世紀内の年の部分を 94 未満 (場合によっては 95 未満)に設定しようとすると、実際には 94(または 95)が設定されてしまう。このようなマシンでは、hwclock は年を 1999 以降に設定できず、またクロックの値を通常のように正しい値としては用いることができない。

  本当は BIOS を更新するのが絶対に良いが、そうできない場合にこの問題を補償するには、これらのマシンを用いるとき、常に--badyear オプションを指定すること。hwclock は、自分が頭のイカれたクロックを扱っていることを知ると、ハードウェア・クロックの年の部分を無視し、adjtime ファイルの「最終時計合わせ日付」から現在の年を推定しようとする。この動作を行わせたい場合には、hwclock --set または hwclock--systohc を少なくとも年に一回は実行するほうが良いだろう。

  hwclock は、ハードウェア・クロックの読み込み時には年の値を無視するが、設定時には年も設定する。これは 1995、1996、1997、1998 のいずれかとなり、閏年のサイクルに合う年が選択される。このようにして、ハードウェア・クロックに閏日を挿入させるのである。繰り返すが、ハードウェア・クロックを設定せずに一年以上動作させつづけると、この機能が動作せず、一日を失うことになる。

  ハードウェア・クロックが 1994 または 1995 になっていると、hwclock は--badyear が必要ではないか、という警告を発する。

- --srm

このオプションは--epoch=1900と等しく、SRMコンソールのAlphaで最も一般的な紀元年を指定するのに使われる。

- --arc

  このオプションは--epoch=1980と等しく、ARCコンソールのAlphaで最も一般的な紀元年を指定するのに使われる(ただしRuffiansでは1900を紀元年にしている)。

- --jensen、--funky-toy

  これら2つのオプションは、使っているAlphaマシンがどのような種類のものであるか指定する。Alpha以外では無効だし、Alphaでも実際には指定しなくても良いだろう。hwclockは自分が動作しているマシンの種類を自分で決定できるはずである(最低でも/procがマウントされていれば)。(hwclock が正しく動作しないことがわかった場合には、メンテナに連絡して、あなたのシステムを自動検知できるようにプログラムを改良できないか相談してみてほしい。'hwclock --debug'と'cat /proc/cpuinfo'の出力が役立つかもしれない。)

  --jensenは、Jensenモデルを動作させていることを意味する。

  --funky-toyは、そのマシンでは時間の遷移の検知にハードウェア・クロックのUIPビットではなくUFビットが使われていることを意味する。オプション名の"Toy"は、マシンの"Time Of Year" 機能からとったものである。

- --test

  実際のハードウェア・クロックの更新(およびそれに類する)作業をのぞき、すべての動作を行う。このオプションは--debugと組み合わせるとhwclockの動作を理解する上で有用であろう。

- --debug

  hwclockが内部で行っている動作に関して大量の情報を表示する。一部の機能は複雑であるが、この出力はプログラムの動作を理解する上で助けになるだろう。

注意 (Linuxシステムにおける時計)

Linuxシステムには主要な時計が2つ存在する。

- ハードウェア・クロック

  これはCPU内部で動作しているすべてのコントロールプログラムから独立しており、マシンの電源がOFFのときにも動作している。

  ISAシステムでは、このクロックはISA規格の一部として定義されている。コントロールプログラムはこの時計に対して1秒単位で読み書きできるが、秒針の変化を検出することもできるので、実際には仮想的に無限大の精度を持っていることになる。

  この時計は一般にハードウェア・クロック、リアルタイム・クロック、RTC、BIOS クロック、CMOSクロックなどと呼ばれる。hwclockでは「ハードウェア・クロック(原文ではHardware Clock)」を用いる。他の名前は不正確だったり誤解のもとになるからである。

- システム・クロック

これは Linux カーネルの内部に存在している時計で、タイマ割り込みによって駆動されている(ISA システムでは、タイマ割り込みは ISA 標準の一部である)。すなわち Linux が起動している間しか動作しない。システム時刻は UTC 1970/01/01 00:00:00 からの経過秒数である(より簡単に言えば 1969 年終了後の経過秒数である)。しかしシステム時刻は整数ではなく、仮想的に無限大の精度を持っている。

Linux ではシステム・クロックがすべての基準となる時計である。ハードウェア・クロックの基本的な役割は、システムが動いていない間にも時計を動かしつづけることである。Linux システムは起動時に一度だけハードウェア・クロックを参照し、システム・クロックを設定する。その後はハードウェア・クロックは用いない。ISA システムの設計対象であった DOS においては、ハードウェア・クロックがただ一つの実時間時計であることに注意すること。

システム・クロックには不連続が存在してはならない。これはシステムが走っている間に date(1L)プログラムを実行して時計を合わせるような場合でも同様である。一方ハードウェア・クロックには、システムの実行中にでも何を行ってもよい。次回 Linux が起動したときに、ハードウェア・クロックからこの調整された時間が使用される。システムが走っている間にシステム・クロックをスムースに修正するには adjtimex(8)を用いることもできる。

Linux カーネルは、システムのローカルなタイムゾーンという概念を持っている。しかし注意してほしい --「カーネルが自分をどのタイムゾーンにいると思っているか」など、誰も気にしていないのである。代わりに、タイムゾーンに関するプログラム(おそらくローカルな時間を表示しようとしているもの)は、ほぼ間違いなく従来用いられてきた方法でタイムゾーンを決定する。つまり TZ 環境変数や/usr/local/timezone ディレクトリを、tzset(3)で説明されているようなやり方で参照するのである。しかしカーネルのタイムゾーンの値を見るプログラムも存在するし、カーネルの周辺部分(ファイルシステムなど)もこちらを参照する。vfat ファイルシステムなどがそうである。カーネルのタイムゾーンの値が間違っていると、vfat ファイルシステムはファイルのタイムスタンプの設定・取得を間違ってしまう。

hwclock は--hctosys オプションでシステム・クロックをセットするとき、カーネルのタイムゾーンも TZ や/usr/local/timezone の値に設定する。

タイムゾーンの値は実際には 2 つの部分からなる。1) tz_minuteswest フィールド:(DST でない)ローカルタイムが UTC から何分遅れているかを表す。2) tz_dsttime: 夏時間(DST)の形式を表し、現在地の現在時刻に影響する。この 2 番目のフィールドは Linux では用いられず、常に 0 となる。(settimeofday(2)も参照のこと。)

hwclock がハードウェア・クロックへアクセスする方法

hwclock はハードウェア・クロック時刻の取得や設定に、いろいろな方法を用いる。もっとも普通のやり方は、デバイススペシャルファイル/dev/rtc に対して I/O を行う方法である。しかしこの方法が常に利用で

きるとは限らない。そもそも rtc ドライバが Linux へ追加されたのは比較的最近のことである。古いシステムには存在しない。DEC Alpha で動作する rtc ドライバもあるが、このドライバが使えない Alpha マシンもたくさんあるようである(症状としては hwclock がハングする)。

古いシステムでは、ハードウェア・クロックへのアクセス方法はシステムのハードウェアに依存している。

ISA システムでは、hwclock は時計を構成していた「CMOS メモリ」のレジスタに直接アクセスすることができた(ポート 0x70 と 0x71 に I/O を行う)。これを行うには hwclock の実効ユーザーID がスーパーユーザーでなければならない。(Jensen Alpha の場合は、このような I/O 命令を hwclock に実行させることはできない。したがってこの場合はデバイススペシャルファイル/dev/port が用いられる。これは I/O サブシステムへの低レベルインターフェースのほとんどを与えるものである。)

これは時計にアクセスする方法としては実に情けない方法である。ユーザー空間のプログラムでは、このように直接 I/O を叩いたり、割り込みを禁止したりすることは通常想定されていないのだから。hwclock でこれが使えるようにしてあるのは、古い Linux カーネルで ISA マシンを使う場合には、これが唯一の方法だからである。

m68k システムでは、hwclock はコンソールドライバとデバイススペシャルファイル/dev/tty1 を通して時計にアクセスすることができる。

hwclock は/dev/rtc を用いようとする。この機能を持たないカーネル向けにコンパイルされていたり、/dev/rtc をオープンできない場合には、hwclock は他の方法を(可能であれば)試そうとする。ISA や Alpha のマシンでは、/dev/rtc を試さずに、最初から hwclock に CMOS レジスタを直接操作するように強制することもできる。これには--directisa オプションを指定する。

時刻合わせ機能

通常ハードウェア・クロックはそれほど正確なものではない。しかし、その「不正確さ」は完全に予測できるものである。すなわち、時計は一日あたり同じ時間だけ進む(あるいは遅れる)のである。これを規則的なずれ(systematic drift)と呼ぶことにする。hwclock の時刻合わせの機能は、この規則的なずれに対応する補正量を求め、適用するものである。

以下に動作原理を述べる。hwclock は/etc/adjtime というファイルを管理し、そこに履歴情報を保管する。このファイルを adjtime ファイルと呼ぶ。

adjtime ファイルがない状態から話をはじめる。hwclock --set コマンドを用いてハードウェア・クロックを現在の正しい値に合わせたとする。このとき hwclock は adjtime ファイルを作成し、そこに現在の時刻を「最後に時計合わせ(calibration)が行われた時刻」として記録する。五日後に時計は 10 秒進んだとし、それを修正するために再び hwclock--set が実行されたとする。hwclock は adjtime ファイルを更新し、現在の時刻を最後に時計合わせが行われた時刻として記録、同時に 2 秒/日という値を規則的なずれの値として

記録する。24 時間が経過したときに hwclock --adjust コマンドを実行すると、hwclock は adjtime ファイルを参照し、放っておかれた時計は一日に 2 秒進むこと、時計はちょうど一日だけ放置されていたことを読みとる。そこで hwclock はハードウェア・クロックから 2 秒を差し引き、現在の時刻を時計の補正(adjustment)が行われた時刻として記録する。さらに 24 時間が経過したときに hwclock --adjust を実行すれば、hwclock はまた同じことを行う。つまり 2 秒を差し引き、現在の時刻を adjtime ファイルに書き込む。

(--set または--systohc を用いて)時計を合わせるごとに、hwclock は規則的なずれを再計算する。このときには、最後に時計合せが行われた時点からの経過、途中で行われた補正で用いられていたずれの量、最後に補正を行った時刻からの経過時間などが参照される。hwclock が時計を設定するときには、常に小さなずれが生じる可能性がある。これが 1 秒に満たない場合には、時計の補正量からは切り捨てられる。後に再び補正を行う際に、このずれが蓄積して 1 秒を越えていれば、その分はその時に補正される。

システムの起動時に(あるいはシステムの動作中に cron で定期的に) hwclock --hctosys を行う時には、常にその前に hwclock --adjust を行うと良いだろう。

adjtime ファイルは、当初は修正量(adjustments)だけを目的としていたためにこの名前がつけられたが、現在では他の情報も書き込まれており、hwclock が一度起動され、次に起動されるまでにその情報を保持する。

adjtime は ASCII ファイルであり、フォーマットは以下の通り:

一行目は三つの数値からなり、それぞれ空白で区切られる:

- 一日あたりに生じる時刻ずれを秒で表したもの(浮動小数点型 10 進)
- 最後に時計合わせあるいは補正を行った時刻を 1969 UTC からの経過秒数で表したもの(10 進整数)
- ゼロ(clock(8)との互換性のためのもの)

二行目: 数値が一つ:最後に時計を合わせた時刻を 1969 UTC からの経過秒数で表したもの。時計合わせが一度もされていなかったり、以前の時計あわせに問題があった(例えばその時計あわせ以降にハードウェア・クロックの時刻が不正なことがわかったとか)場合には 0 が入る。これは 10 進の整数である。

三行目: "UTC"または"LOCAL"。ハードウェア・クロックが協定世界時かローカルタイム化を示す。この値は hwclock にコマンドラインを指定すればいつでも上書き可能である。

以前 clock(8)で使っていた adjtime ファイルは hwclock でもそのまま使うことができる。

カーネルによるハードウェアクロックの自動合わせ

ハードウェアクロックを正しい値に同期させるのに、別法が取れるようなシステムもある。Linux カーネル

には、11 分ごとにシステムクロックをハードウェアクロックにコピーするようなモードが存在する。これは、何らかの洗練された方法(ntp など)でシステムクロックを同期できている時には、よいモードであろう。(ntp とは、ネットワークのどこかにあるタイムサーバーか、システムに付属した電波時計にシステム・クロックを同期させる手法である。RFC1305 を見よ。)

このモード(「11 分モード」と呼ぶ)は、何かによって有効にされるまではオフになっている。例えば ntp デーモンである xntpd はこのモードを有効にできるもののひとつである。オフにするのも何かを実行すればよく、例えば hwclock--hctosys を実行して、システム・クロックを古い方法で設定すれば、11 分モードはオフになる。

モードがオンかオフかを調べるには、adjtimex--print コマンドを実行して"status"の値を見ればよい。この数値の第 64 ビットが(2 進数表示で)0 ならば、11 分モードはオンになっている。それ以外の場合はオフである。

システムが 11 分モードで動作している場合に hwclock --adjust や hwclock --hctosys を実行してはならない。システムをおかしくしてしまう。hwclock --hctosys を起動時だけに用いるならかまわない。これを用いれば、システム・クロックが外部の値に同期して 11 分モードが開始されるまで、システムクロックを妥当な値にできる。

ISA ハードウェア・クロックの「世紀値(Century value)」

その手の標準の中には、ISA マシンの CMOS 50 バイト目を、現在の世紀の指標として定義しているものがある。hwclock は、このバイトの読み書きを行わない。なぜならこのバイトをそのようには利用していないマシンが存在するし、いずれにしてもこれは実際には必要ないからである。年の世紀の部分を使えば、現在の世紀を特定するには充分である。

もしこの CMOS の世紀バイトの利用ルーチンを開発した(したい)方がいたら、hwclock のメンテナに連絡してほしい。オプションを付加することは望ましいことであるから。

このセクションが意味を持つのは、ハードウェア・クロックに"directISA"によってアクセスしている場合だけであることに注意。

環境変数

TZ

ファイル

- /etc/adjtime
- /usr/share/zoneinfo/(古いシステムでは /usr/lib/zoneinfo)

- /dev/rtc
- /dev/port
- /dev/tty1
- /proc/cpuinfo

関連項目

adjtimex(8)、date(1)、gettimeofday(2)、settimeofday(2)、crontab(1)、tzset(3)

6.2.5. 監査の閲覧

6.2.5.1 ファイルを連結して出力する（cat）

書式

cat [-benstuvABET] [--binary] [--number] [--number-nonblank] [--show-all] [--show-ends] [--show-nonprinting] [--show-tabs] [--squeeze-blank] [FILE...]

cat [--help] [--version]

説明

cat は指定したファイルそれぞれの内容を標準出力へ書き出す。FILE が一つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

オプション

- -b、--number-nonblank

  空白でない行に番号を付ける。初めの行を 1 行目とする。
- -e

  '-vE' と同じ。
- -n、--number

  すべての行に番号を付ける。初めの行を 1 行目とする。
- -s、--squeeze-blank

  連続した空白行を、1 つの空白行にまとめる。
- -t

  '-vT' と同じ。
- -u

  何もしない。Unix との互換性のために存在する。
- -v、--show-nonprinting

<LFD>と<TAB>とを除く制御文字を '^' 表記を使って表示する。高位ビットがセットされている文字は、前に 'M-' を置いて表わす。

- -A、--show-all
  '-vET' と同じ。
- -B、--binary
  DOS プラットフォームのみ。ファイルの読み書きをバイナリモードで行う。バイナリ/テキストモードについては以下の記述を参照のこと。
- -E、--show-ends
  各行の最後に '$' を表示する。
- -T、--show-tabs
  <TAB>文字を '^I' と表示する。
- --help
  標準出力に使用方法のメッセージを出力して正常終了する。
- --version
  標準出力にバージョン情報を出力して正常終了する。

DOS のテキスト/バイナリモード

cat はデフォルトではテキストモードを用い、標準出力がファイルまたはパイプにリダイレクトされたときにはバイナリモードを用いる。この振舞いはオプションによって変更でき、--binary や--show-nonprinting を指定すればバイナリモードに、--number-blank や--show-ends や--number を指定すればテキストモードになる。

テキストモードでは文字変換が行われる(例えば<CR>を<CR><LF>へ、など)。したがって cat を使ったファイルコピーには適さない。オリジナルの内容を保存しないからである。

6.2.5.2 指定した桁をファイルの各行から削除する (colrm)

書式

```
colrm [start [stop]]
```

解説

colrm は、指定された桁をファイルの各行から削除します。1 桁は 1 文字分に相当します。ファイルは標準入力から読み込まれ、結果は標準出力に書き出されます。

start だけが与えられた場合は、start 桁目より前のすべての桁が表示されます。start と stop の両方が指定された場合は、start 桁目より前のすべての桁と、stop 桁目より後のすべての桁が表示されます。桁の

番号づけは 0 ではなく 1 から始まります。

タブ文字を読むと、その桁位置の次に来る 8 の倍数の桁まで桁位置を進めます。バックスペース文字を読むと、桁位置を 1 戻します。

関連項目

awk(1)、column(1)、cut(1)、paste(1)

6.2.5.3 入力を複数カラムに分けて整形する（column）

書式

```
column [-tx] [-c columns] [-s sep] [file ...]
```

説明

column は、入力を複数カラムに分けて整形します。列より先に行を埋めます。file が指定されていれば file を、指定されていなければ標準入力を処理します。空行は無視されます。

オプション

- -c
  表示の幅を columns にしてフォーマットします。
- -s
  -t オプションを使う時に、入力行をカラムに分ける区切り文字(複数でも良い)を指定します。
- -t
  入力行のカラム数を判定し、表を作ります。カラムの区切りは、-s オプションで指定された文字か、指定されていない場合は空白文字です。画面表示をきれいに整形するのに便利です。
- -x
  行を埋める前に列を埋めます。

環境変数

- COLUMNS
  他に情報が得られない時に、画面の横幅を指定します。

使用例

```
(printf "PERM LINKS OWNER GROUP SIZE MONTH DAY " ; ¥
printf "HH:MM/YEAR NAME¥n" ; ¥
```

```
ls -l | sed 1d) | column –t
```

関連項目

colrm(1)、ls(1)、paste(1)、sort(1)

6.2.5.4 ソート済みの 2 つのファイルを行ごとに比較する（comm.）

書式

```
comm [-123] FILE1 [FILE2]
comm [--help] [--version]
```

説明

comm は二つの入力ファイルを読み込み、共通な行および共通でない行をそれぞれ表示する。一方の FILE が省略されると、その分は標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

オプションが与えられない場合、comm は 3 列からなる出力を生成する。第 1 列には FILE1 だけにしか含まれない行を、第 2 列には FILE2 だけにしか含まれない行を、そして第 3 列には両方のファイルに共通に含まれている行をそれぞれ出力する。列は<TAB>区切りである。

オプション

- -1
  第 1 列の出力をしない。
- -2
  第 2 列の出力をしない。
- -3
  第 3 列の出力をしない。
- --help
  標準出力に使用方法のメッセージを出力して正常終了する。
- --version
  標準出力にバージョン情報を出力して正常終了する。

返り値

他の比較ユーティリティと異なり、comm の返す終了ステータスは比較結果に依存しない。通常終了をすると、comm は終了ステータスとして 0 を返し、エラーがあると 0 以外のステータスを返す。

6.2.5.5 ファイルを文脈ベースで分割する (csplit)

書式

```
csplit [-kqsz] [-b SUFFIX] [-f PREFIX] [-n DIGITS] [--digits=DIGITS]
[--elide-empty-files] [--keep-files] [--prefix=PREFIX] [--quiet] [--silent]
[--suffix=SUFFIX] [FILE]

csplit [--help] [--version]
```

説明

csplit は FILE の各セクション(section)の内容を持つ 0 個以上の出力ファイルを作成する。FILE が一つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

出力ファイルの内容は PATTERN 引数によって決まる。PATTERN 引き数で指定される行が入力ファイルに存在しない場合はエラーとなる(例えば、与えられた正規表現にマッチする行が残っていない場合など)。すべての PATTERN がマッチしおわったら、入力の残りは最後の出力ファイルにコピーされる。

出力ファイルのファイル名は二つの部分からなっている。前半部のデフォルト名は 'xx' である。デフォルトでは後半部は 2 桁の 10 進数で、'00' から '99' まで順にカウントアップされて行く。いずれの場合でも、出力ファイルを名前でソートして順に結合させると、入力ファイルが得られるようになっている。

デフォルトでは、csplit はそれぞれの出力ファイルを作成した後に、それに書き出したバイト数を表示する。

csplit がエラーになったり、hangup、interrupt、quit、terminate 各シグナルを受け取った場合には、デフォルトではそれまでに作られた出力ファイルは消去される。

オプション

- -b SUFFIX、--suffix=SUFFIX

  SUFFIX を出力ファイル名の後半部に用いる。SUFFIX には、printf(3)形式の変換文字列を一つだけ指定しなければならない。書式指定フラグ、文字列の幅、精度指定なども指定できる。変換指定文字列は整数を可読なかたちで出力するものでなければならない。書式は整数値を可読な形式に変換するものでなければならない。したがって使えるのは 'd'、'i'、'u'、'o'、'x'、'X' に限られる。

  suffix 文字列はすべて(現在の出力ファイルが何番目かを示す数値と共に)sprintf(3)関数に渡され、それぞれの出力ファイルに応じた名前が順番に出力される。このオプションが指定されると、--digits オプションは無視される。

- -f PREFIX、--prefix=PREFIX

  出力ファイル名の前半部を PREFIX にする。

- -k、--keep-files

  エラーが起こったとき、それまでに作成した出力ファイルを消去しないようにする。

- -n DIGITS、--digits=DIGITS

  出力ファイル名の数値部分の桁数をデフォルト値の 2 から DIGITS に変更する。

- -q、-s、--quiet、--silent

  出力ファイルのサイズを表示しない。

- -z、--elide-empty-files

  大きさ 0 の出力ファイルを作らないようにする。(セクション区切りが入力の最初の行にきた場合、このオプションを指定していなければ対応する出力ファイルの大きさは 0 になる)。なお出力ファイルの順番を示す番号は、このオプションが指定された場合でも、常に 0 から連続してカウントされる。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

PATTERN 引数

- N

  N には正の整数を指定する。出力ファイルを作成し、入力ファイルの内容を行番号が N になるまで書き込む(ただし line の行は書き込まない)。この後に繰り返し回数を指定すると、それぞれ N 行分の入力ファイルの内容を含む出力ファイルを作成する。

- /REGEXP/[OFFSET]

  出力ファイルを作成し、入力ファイルのうち REGEXP にマッチする行の前までの内容を書き込む(マッチ行は含まない)。OFFSET を付加することもでき、これは '+' または '-' に正の整数を続けて指定する。OFFSET が指定されるとマッチ行に OFFSET の値を増減した行までが出力される。入力ファイルの残りは次のセクションの入力として利用される。

- %REGEXP%[OFFSET]

  直前の形式と同様だが、出力ファイルが作成されない。したがって入力ファイルの該当するセクションは無視されることになる。

- {REPEAT-COUNT}

  直前のパターンを REPEAT-COUNT の回数だけ余計に繰りかえす。REPEAT-COUNT には正の整数かアスタリスクが指定できる。アスタリスクの場合は、入力ファイルが終わるまで直前のパターンを繰り返す。

6.2.5.6 各行から選択した部分を表示する(cut)

書式

```
cut [-ns] [-b BYTE-LIST] [-c CHARACTER-LIST] [-d DELIM] [-f FIELD-LIST]
[--bytes=BYTE-LIST]  [--characters=CHARACTER-LIST] [--delimiter=DELIM]
[--fields=FIELD-LIST] [--only-delimited] [FILE...]

cut [--help] [--version]
```

説明

cut は、与えられた FILE それぞれから、各行の一部を選択して標準出力に書き出す。FILE が一つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

BYTE-LIST、CHARACTER-LIST、FIELD-LIST には、一つ以上の数値か範囲(ダッシュで接続された 2 つの数値)をコンマで区切って指定する。バイト・文字・フィールドの番号は 1 から振られる。不完全な範囲を指定することもできる: '-M' は '1-M' という意味に、'N-' は 'N' 番目から行の最後のフィールドまでという意味になる。

オプション

- -b BYTE-LIST、--bytes=BYTE-LIST

  BYTE-LIST にリストされた位置の各バイトだけを表示する。タブやバックスペースもほかの普通の文字と同じように 1 バイトとして扱う。
- -c CHARACTER-LIST、--characters=CHARACTER-LIST

  CHARACTER-LIST にリストされた位置の各文字だけを表示する。これはいまのところ-b と同じだが、国際化すると異なる動作となるだろう。タブやバックスペースもほかの普通の文字と同じように 1 文字として扱う。
- -d DELIM、delimiter=DELIM

  -f と一緒に用いると、フィールドの区切り文字として DELIM の先頭の文字を使う(デフォルトは <TAB>)。
- -f FIELD-LIST、--fields=FIELD-LIST

  FIELD-LIST にリストされた各フィールドだけを表示する。フィールドの区切りはデフォルトでは <TAB>である。
- -n

  マルチバイト文字を分割しない(現在は効果がない)。
- -s、--only-delimited

  -f と一緒に用いると、フィールドの区切り文字を含まない行を表示しない。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.7 2 つのファイル間の違いを探す (diff)

書式

```
diff [-abcdefhilnpqrstuwyBEHNPT] [-LINES] [-x PATTERN] [-C LINES] [-D NAME] [-F REGEXP]
[-I REGEXP] [-L LABEL]        [-S  FILE] [-U LINES] [-W COLUMNS] [-X  FILE] [--binary]
[--brief] [--changed-group-format=fORMAT] [--context[=LINES]] [--diff-program=PROGRAM]
[--ed] [--exclude=PATTERN] [--exclude-from=FILE] [--expand-tabs] [--forward-ed]
[--from-file=FILE] [--horizon-lines=LINES] [--ifdef=NAME] [--ignore-all-space]
[--ignore-blank-lines] [--ignore-case] [--ignore-file-name-case]
[--ignore-matching-lines=REGEXP] [--ignore-space-change] [--ignore-tab-expansion]
[--inhibit-hunk-merge] [--initial-tab] [--label=LABEL] [--left-column]
[--line-format=FORMAT] [--minimal] [--new-file] [--new-group-format=FORMAT]
[--new-line-format=FORMAT] [--no-ignore-file-name-case] [--old-group-format=FORMAT]
[--old-line-format=FORMAT] [--paginate] [--rcs] [--recursive] [--report-identical-files]
[--sdiff-merge-assist] [--show-c-function] [--show-function-line=REGEXP]
[--side-by-side] [--speed-large-files] [--starting-file=FILE] [--strip-trailing-cr]
[--supress-common-lines] [--text] [--to-file=FILE] [--unchanged-group-format=FORMAT]
[--unchanged-line-format=FORMAT] [--unidirectional-new-file] [--unified[=LINES]]
[--width=COLUMNS] FROMFILE TOFILE diff [-v] [--help] [--version]
```

説明

diff は 2 つのファイルを比較し、それらの違いを記述して出力する。

もっとも簡単な場合は、比較対象のファイルは FROMFILE と TOFILE である。これらのファイルのうちのどちらかは '-' と指定しても良い。この時はそのファイルは標準入力から読み込まれる。

FROMFILE がディレクトリで TOFILE がディレクトリではない場合、diff はディレクトリ FROMFILE にある、ファイル名が TOFILE のファイルを比較対象にする(逆も同様)。ディレクトリでない方の指定を '-' にすることはできない。

両方がディレクトリのときは、diff は双方のディレクトリにある、対応するファイルをアルファベット順にそれぞれ比較する。比較は再帰的には行われないが、-r または--recursive オプションを指定すれば再帰的になる。diff はディレクトリの内容そのものをファイルのように比較することはしない。フルパス指定のファイルは '-' であってはならない。なぜなら標準入力には名前がないので、「同名のファイル」という概念が使えないからである。

--from-file=FILE オプションが与えられると、任意の数のファイル名を与えることができ、それぞれのファイルは FILE と比較される。同様に、--to-file=FILE オプションが与えられると、それぞれ指定されたファイ

ルは FILE と比較される。[訳注:2.7 にはこれらのオプションは存在しない]

オプション

- -LINES

  異なっている部分の前後 LINES 行(整数)分のコンテクストを表示する。このオプションは出力形式自体の指定は行わない。したがって-c や-u オプションと一緒に用いると、なんの効果も持たない。このオプションは obsolete である。patch(1)が正しく動作するには、少なくとも 2 行のコンテクストが必要である。

- -a、--text

  ファイルがテキストには見えないような場合でも、全てのファイルをテキストとみなして 1 行ずつ比較を行う。

- -b、--ignore-space-change

  スペースの数だけが違う場合には違いを無視する。不完全な行は無視される。

- -c context

  出力形式を用いる。

- -d、--minimal

  アルゴリズムを変更し、より小さな差分が生成できるようにする。これを使うと diff は遅くなる(非常に遅くなる場合もある)。

- -e、--ed

  ed(1)のスクリプト形式で出力する。

- -f、--forward-ed

  ed のスクリプトと一見同じような出力をする。しかし出力に現れる順序が異なる[訳注:したがって ed では使えない]。

- -h

  現在は何も効果を持たない。Unix との互換性のために存在している。これは推奨されない。

- -i、--ignore-case

  英大文字と小文字の違いを無視する。

- -l、--paginate

  出力を pr(1)に通してページ付けを行う。

- -n、--rcs

  RCS 形式の diff を出力する。-f と似ているが、それぞれのコマンドは処理する行数を指定する。

- -p、--show-c-function

  変更がどの C 関数で行われたのかを表示する。'-F' ^[_a-zA-Z$]' ' と同じ。

- -q、--brief

  ファイルが違うかどうかだけを報告する。違いの詳細は報告しない。

- -r、--recursive

  ディレクトリを比較するとき、見付かったサブディレクトリをすべて再帰的に比較する。

- -s、--report-identical-files

  2 つのファイルが同じだったときも報告する。

- -t、--expand-tabs

  入力ファイルでのタブによる位置あわせを保存するため、出力のタブをスペースに展開する。

- -u

  unified 出力形式を用いる。

- -w、--ignore-all-space

  行を比較するときスペースを無視する。不完全な行は無視される。

- -x PATTERN、--exclude=PATTERN

  ディレクトリを比較するとき、ファイル名の base 部が PATTERN にマッチするファイルやサブディレクトリを無視する。

- -y、--side-by-side

  side-by-side 出力形式を用いる。

- -B、--ignore-blank-lines

  空行を挿入・削除するだけの変更を無視する。

- -C LINES、--context[=LINES]

  context 出力形式を用い、LINES 行(整数値)のコンテクストを表示する。LINES が与えられなければ 3 行表示する。patch が正しく動作するためには、少なくとも 2 行のコンテクストが必要であることが多い。

- -D NAME、--ifdef=NAME

  if-then-else 形式でマージされた出力を行い、プリプロセッサの条件マクロに NAME を用いる。

- -E、--ignore-tab-expansion

  タブ展開によるスペースの変更を無視する。

- -F REGEXP、--show-function-line=REGEXP

  context 形式と unified 形式において、各差分 hunk(テキストブロック)に対し、その前方で REGEXP にマッチした最後の行の一部を表示する。

- -H、--speed-large-files

  小さな変更が大量にあるような大きなファイルを高速に扱うために、ヒューリスティックな手法を用いる。短縮形式-H は推奨されなくなった。

- -I REGEXP、--ignore-matching-lines=REGEXP

  REGEXP にマッチするような行を挿入・削除するだけの変更を無視する。

- -L LABEL、--label=LABEL

  context 形式と unified 形式のヘッダに、ファイル名ではなく LABEL を用いる。短縮形式-L は推奨されなくなった。

- -N、--new-file

  ディレクトリを比較する際、片方のディレクトリにのみファイルが存在していたら、もう片方のディレクトリには同名の空っぽのファイルがあるように動作する。

- -P、--unidirectional-new-file

  ディレクトリを比較する際、2 番目のディレクトリにのみファイルが存在していたら、1 番目のディレクトリには同名の空っぽのファイルがあるように動作する。短縮形式-P は推奨されなくなった。

- -S FILE、--starting-file=FILE

  ディレクトリを比較する際、FILE から始める。中断した比較を続行する際に利用できる。

- -T、--initial-tab

  normal 形式や context 形式で、テキストの前にスペースでなくタブを出力する。こうすると行中でのタブによる桁揃えが普通に見える。

- -U LINES、--unified[=LINES]

  unified 出力形式を用い、LINES 行(整数値)のコンテクストを表示する。LINES が与えられなければ 3 行表示する。patch が正しく動作するためには、少なくとも 2 行のコンテクストが必要であることが多い。

- -W COLUMNS--width=COLUMNS

  side-by-side 形式で、出力の幅を COLUMNS にする。

- -XFILE、--exclude-from=FILE

  ディレクトリを比較する際、ファイル名の base 部が FILE のパターンのいずれかにマッチするファイルやサブディレクトリを無視する。

- --binary

  データをバイナリモードで読み書きする(Linux やその他の POSIX ホストでは意味なし)。

- --changed-group-format=FORMAT

  if-then-else 形式で、両方のファイルで異なる行グループの出力に FORMAT を用いる。

- --diff-program=PROGRAM

  ファイルの比較するために diff と互換性のある外部プログラム PROGRAM を用いる。

- --from-file=FILE

  FILE を各オペランドと比較する(FILE はディレクトリでも良い)。[訳注:2.7 にはこのオプションは存在しない]

- --horizon-lines=LINES

  差分をもっともコンパクトに出力するために、違う部分の前後にある共通部分のそれぞれ LINES 行を捨てずに保存する。

- --ignore-file-name-case

  ファイルを比較する際にファイル名の大文字小文字を無視する。そのため‘‘foo’’と‘‘Foo’’は同じとされるので、互いに比較される。

- --inhibit-hunk-merge

  隣接する hunk の境界を移動して hunk をマージする動作を行わない。

- --left-column

  side-by-side 形式で、共通な行は左側の列にしか表示しない。

- --line-format=FORMAT

  if-then-else 形式で、全ての入力行の出力に FORMAT を用いる。

- --new-group-format=FORMAT

  if-then-else 形式で、2 番目のファイルだけにある行グループの出力に FORMAT を用いる。

- --new-line-format=FORMAT

  if-then-else 形式で、2 番目のファイルだけにある行の出力に FORMAT を用いる。

- --no-ignore-file-name-case

  ファイルを比較する際に、ファイル名の大文字小文字を考慮する。そのため ‘‘foo’’ と ‘‘Foo’’ は同じものとされない。--ignore-file-name-case を参照すること。

- --old-group-format=FORMAT

  if-then-else 形式で、1 番目のファイルだけにある行グループの出力に FORMAT を用いる。

- --old-line-format=FORMAT

  if-then-else 形式で、1 番目のファイルだけにある行の出力に FOR-MAT を用いる。

- --sdiff-merge-assist

  sdiff(1)用に追加情報を表示する。sdiff が diff を実行するときにこのオプションを用いる。通常のユーザーがこのオプションを直接指定する場合はあまり想定されていない。

- --strip-trailing-cr

  行末の CR を取り除く。行末のマーカとして CRLF を使うシステムの出力を処理するときに役立つ。

- --suppress-common-lines

  side-by-side 形式で共通な行を表示しない。

- --unchanged-group-format=FORMAT

  if-then-else 形式で、両方のファイルに共通な行グループの出力に FORMAT を用いる。

- --unchanged-line-format=FORMAT

  if-then-else 形式で、両方のファイルに共通な行の出力に FORMAT を用いる。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- -v、--version

  diff のバージョン番号を出力する。

出力の形式

context 形式

context 出力形式は 2 行のヘッダからはじまる。これは以下のようなものである:

```
*** FROMFILE FROMFILE-MODIFICATION-TIME
--- TOFILE TOFILE-MODIFICATION-TIME
```

-L LABEL を用いるとヘッダの内容は変化する。次に来るのは hunk(テキストブロック)である。繰り返し登場することもある。context 形式の hunk は以下のようなものである:

```
***************
*** FROMFILE-LINE-RANGE ****
FROMFILE-LINE
FROMFILE-LINE...
--- TOFILE-LINE-RANGE ----
TOFILE-LINE
TOFILE-LINE...
```

異なっている行の周辺のコンテクストの各行には、先頭に 2 つのスペースがついている。2 つのファイル間で異なっている行には、先頭に表示文字とスペースがつく。

- !

  2 つのファイル間で変更された部分の各行。もう一方のファイルの対応する部分も '!' でマークされて表示される。

- +

  2 つめのファイルで「挿入された」行。1 つめのファイルに対応部分はない。

- -

  1 つめのファイルで「削除された」行。2 つめのファイルに対応部分はない。

hunk の全ての変更が挿入であれば、FROMFILE 各行は省略される。また全ての変更が削除であれば、TOFILE 各行は省略される。

unified 形式

```
--- FROMFILE FROMFILE-MODIFICATION-TIME
+++ TOFILE TOFILE-MODIFICATION-TIME
```

-L LABEL を用いるとヘッダの内容は変化する。次に来るのは hunk(テキストブロック)である。繰り返し登場することもある。それぞれの hunk はファイルの異なっている 1 つ 1 つの部分を示している。unified 形式の hunk は以下のようなものである:

```
@@ FROMFILE-RANGE TOFILE-RANGE @@
LINE-FROM-EITHER-FILE
LINE-FROM-EITHER-FILE...
```

両者で共通な行の前には 1 つのスペースが置かれる。異なる行の前には以下の表示文字が置かれる:

- +

  ここで 1 つめのファイルに行の追加が行われた。

- -

  ここで 1 つめのファイルから行の削除が行われた。

side-by-side 形式

ファイルは 2 列に表示され、間に溝(gutter)が置かれる。溝には以下のマーカーのいずれか 1 つが含まれる。

- ' '

  対応する行が共通である。つまり、両方の行が同一であるか、違いが--ignore オプションのいずれかによって無視された。

- |

  対応する行が異なる。両方とも完全か、両方とも不完全かである。

- <

  ファイルは異なり、1 番目のファイルにだけこの行が含まれている。

- >

  ファイルは異なり、2 番目のファイルにだけこの行が含まれている。

- (

  1 番目のファイルにだけこの行が含まれているが、違いは無視される。

-)

  2 番目のファイルにだけこの行が含まれているが、違いは無視される。

- ¥

  対応する行が異なる。1 番目の行だけに行末の改行がない。

- /

  対応する行が異なる。2 番目の行だけに行末の改行がない。通常出力行は、そこに含まれる行の末尾に改行がない場合に限って改行されない。しかし、出力行が 2 行の差異を表している場合には、片方だけに改行がない場合もあり得る。この場合出力行には改行が出力されるが、溝のマークは 1 番目の行末に改行がなければ '¥' に、2 番目の行末に改行がなければ '/' になる。

side-by-side 形式が一番読みやすいような場合もあるが、限界もある。この形式は通常よりもずっと広い幅の出力を生成し、長すぎる行は折り畳んでしまう。またこの出力では通常よりも文字の整列への依存が大きくなるので、等幅フォントを使っていなかったり、通常と異なるタブストップを使っていたり、印字されない文字を使っていたりすると、出力が非常に見にくくなる。

ed(1)形式

1 つ以上の差分 hunk から構成される。末尾に近い方の変更が先に現れ、行数の変化が以降の ed の行番号解釈に影響しないようになっている。ed 形式の hunk は以下のようなものである:

CHANGE-COMMAND
TO-FILE-LINE
TO-FILE-LINE...
.

edは入力の末尾を表すためにピリオド1つだけの行を用いるので、diffは変更された行のうち、ピリオド1つだけの行をピリオド2つに変更してプロテクトし、続けて2つのピリオドを1つにするedコマンドを書く。ed 形式は改行されていない行を表すことができない。したがって 2 番目のファイルの最終行が変更されていて、かつ改行されていなかった場合、diff はエラーを報告して、改行があるかのように動作する。

変更コマンドには 3 つの種類がある。それぞれ、1 番目のファイルの行番号 (またはコンマ区切り指定の行範囲指定)、続けて変更の種類を示す 1 文字の文字からなる。行番号は、すべてファイルのオリジナルの行番号である。変更コマンドの種類は以下の通り:

- La

  1 番目のファイルの L 行目以降に、2 番目のファイルからテキストを追加する。例えば '8a' は、以降の行をファイル 1 の 8 行目以降に追加する。
- Rc

  1 番目のファイルの行範囲 R を、引き続く行で置き換える。追加と削除の組み合わせでも同じことができるが、こちらのほうがコンパクトである。例えば '5,7c' はファイル 1 の 5 行目から 7 行目をファイル 2 から読み込んだ内容で置き換える。
- Rd

  1 番目のファイルの行範囲 R を削除する。例えば '5,7d' はファイル 1 の 5 行目から 7 行目までを削除する。

diffはedスクリプトのような出力で、かつ各hunkがforward方向(先頭から末尾へ)のような出力も生成できる。コマンドの形式もちょっと変化している。コマンド文字が、修正する行や行範囲の先に来るのである。また、ピリオド1つだけの行を特別扱いすることもしない。ed 形式と同様に、forward ed 形式も改行していない行を表すことはできない。

forward ed 形式はあまり便利ではない。なぜならedもpatchもこの形式のdiffを用いることができないからである。これは主に古いバージョンの diff との互換性のために存在している。

RCS 形式

RCS 出力形式は Revision Control System (RCS)で用いるために特別に設計されたものである。RCS はバージョンやシステムの異なるファイルを扱うための、フリーなプログラムセットである。この形式は forward ed 形式と似ているが、ピリオド 1 つの行や改行していない行の問題を回避しているので、ファイル内容の任意の変更を表すことができる。テキストセクションをピリオド 1 つの行で終わるのではなく、それぞれのコマンドで、適用する行数を指定しているのである。また 'c' コマンドの代わりに 'a' と 'd' の組み合わせを用いている。さらに 2 番目のファイルの末尾が変更されていて、かつ改行で終わっていない

場合には、出力の末尾も改行されない状態で終わる。

IF-THEN-ELSE

C ソース形式

diff を用いて 2 ファイルの C ソースコードをマージすることもできる。この出力形式には、両方のファイルの行がすべて含まれる。両方のファイルに共通な行は一度しか登場しない。異なる部分は C プリプロセッサの指定を用いて分離される。#ifdef NAME または#ifndef NAME, ,BR #else ″, and ″ #endif である。出力をコンパイルするとき、マクロ NAME を定義したり、未定義のままにすることによって、どちらのバージョンを使うかを選択できる。

例えば、‘wait (&s)’というインスタンスを‘waitpid (-1, &s, 0)’に変更し、新旧のファイルを‘--ifdef=HAVE_WAITPID’オプションによってマージすると、影響を受けた部分のコードは以下のようになるだろう:

```
  do {
#ifndef HAVE_WAITPID
  if ((w = wait (&s)) < 0 && errno != EINTR)
#else /* HAVE_WAITPID */
  if ((w = waitpid (-1, &s, 0)) < 0  &&         errno != EINTR)
#endif /* HAVE_WAITPID */
  return w;
   } while (w != child);
```

行グループ形式

行グループ形式を用いると、if-then-else 入力を受け入れる多くのアプリケーションに適した形式を指定できる。例えばプログラミング言語や文書整形言語などが挙げられる。行グループ形式は、似ている行からなる隣接したグループの出力形式を指定する。

例えば、以下のコマンドは TeX ファイル‘old’と‘new’を比較し、old の部分を‘¥begin{em}’-‘¥end{em}’で囲み、new の部分を‘¥begin{bf}’-‘¥end{bf}’で囲んでマージしたかたちで出力する。

```
diff ¥
   --old-group-format='¥begin{em}
%<¥end{em}
' ¥
   --new-group-format='¥begin{bf}
%>¥end{bf}
' ¥
   old new
```

以下のコマンドも上記の例と同じだが、やや記述が多い。デフォルトの行グループ形式も指定しているからである。

```
diff ¥
```

```
   --old-group-format='¥begin{em}
%<¥end{em}
' ¥
   --new-group-format='¥begin{bf}
%>¥end{bf}
' ¥
   --unchanged-group-format='%=' ¥
   --changed-group-format='¥begin{em}
%<¥end{em}
¥begin{bf}
%>¥end{bf}
' ¥
   old new
```

次にもう少し進んだ例を紹介する。これは差分リストを、"plain English"スタイルで行番号を書いたヘッダとともに出力する。

```
diff ¥
   --unchanged-group-format='' ¥
   --old-group-format='-------- %dn line%(n=1?:s) deleted at %df:
%<' ¥
   --new-group-format='-------- %dN line%(N=1?:s) added after %de:
%>' ¥
   --changed-group-format='-------- %dn line%(n=1?:s) changed at %df:
%<-------- to:
%>' ¥
   old new
```

行グループ形式を指定するには、diff を以下のオプションのどれか 1 つを指定して実行する。4 つまでの行グループ形式を指定でき、各指定がそれぞれ行グループ 1 つに対応する。FORMAT にはシェルのメタキャラクタが入っていることが多いので、クォートするべきであろう。

- --old-group-format=FORMAT

  これらの行グループは 1 番目のファイルだけにある行からなる hunk である。デフォルトの old グループ形式は、changed グループ形式が指定されていればそれと同じになる。されていなければ行グループはそのままのかたちで出力される。

- --new-group-format=FORMAT

  これらの行グループは 2 番目のファイルだけにある行からなる hunk である。デフォルトの new グループ形式は、changed グループ形式が指定されていればそれと同じになる。されていなければ行グループはそのままのかたちで出力される。

- --changed-group-format=FORMAT

  これらの行グループは両方のファイルの行からなる hunk である。デフォルトの changed グループ形式は、old グループと new グループの形式を連結したものである。

- --unchanged-group-format=FORMAT

  これらの行グループは両方のファイルに共通の行からなる hunk である。デフォルトの unchanged グループ形式は、行グループをそのままのかたちで出力するものである。

グループ変換

- %<

  1 番目のファイルからの行を意味する。行末尾の改行も含む。各行は old 行形式によって整形される。

- %>

  2 番目のファイルからの行を意味する。行末尾の改行も含む。各行は new 行形式によって整形される。

- %=

  両方のファイルで共通な行を意味する。行末尾の改行も含む。各行は unchanged 行形式によって整形される。

- %%

  '%' を表す。

- %c' C' '

  ここで C は文字 1 文字で、C を表す。C にバックスラッシュやアポストロフィは指定できない。例えば '%c' :' ' はコロンを表し、これは if-then-else 形式の then 部分でもコロンとして解釈される。通常はコロンは then 部分の終わりとして扱われる。

- %c' ¥O' '

  ここで 0 は 1 桁から 3 桁までの 8 進数字であり、8 進のコード 0 に対応する文字を表す。例えば '%c' ¥0' ' はナル文字になる。

- (A=B?T:E)

  A が B に等しい場合は T、等しくない場合は E。A と B はそれぞれ 10 進数の定数か、上記のように解釈される文字 1 つである。この形式指定は A の値が B と等しければ T と等価であり、それ以外の場合は E と等価である。

  例えば '%(N=0?no:%dN)line%(N=1?:s)' は N(new ファイルからのグループの行数)が 0 なら 'no lines' となり、N が 1 なら '1line' となり、それ以外の場合は '%dN lines' となる。

- FN

  ここで F は printf(3)の変換指定で、N は以下の文字のどれかである。「F で整形された N の値」を表す。

  e　old ファイルからのグループの直前の行の行番号。

  f　old ファイルからのグループの最初の行番号。e+1 に等しい。

  l　old ファイルからのグループの末尾の行番号。

  m　old ファイルからのグループの直後の行の行番号。l+1 に等しい。

  n　old ファイルからのグループの行数。l-f+1 に等しい。

  E、F、L、M、N　上記と同様の new ファイルからのグループのもの。

- printf

  変換指定には%d、%o、%x、%X(それぞれ 10 進、8 進、小文字 16 進、大文字 16 進)が使える。

'%' の後には以下のオプションを順に指定できる。'-'(左詰めの指定)、整数(フィールドの最低幅)、ピリオドと数値(数値は省略可;桁数の最小値)である。例えば '%5dN' は new ファイルからのグループの行数を、5 文字幅のフィールドに、printf の"%5d"書式を用いて表示する。

行形式

行形式は、入力から取得された各行を if-then-else 形式の行グループとして出力される際の制御を行う。

例えば、以下のコマンドは、テキストの左に変更表示の 1 文字を表示してテキストを出力する。出力の最初の桁は、削除行では '-'、追加行では '|' となり、変更されなかった行ではスペースとなる。この形式では、改行が必要な部分には改行を入れて出力する。

```
diff ¥
   --old-line-format='-%l
' ¥
   --new-line-format='|%l
' ¥
   --unchanged-line-format=' %l
' ¥
  old new
```

行形式を指定するには、以下のオプションのどれかを用いる。FORMAT にはシェルのメタキャラクタが入っていることが多いので、クォートするべきであろう。

- --old-line-format=FORMAT

  1 番目のファイルからの行だけを整形する。
- --new-line-format=FORMAT

  2 番目のファイルからの行だけを整形する。
- --unchanged-line-format=FORMAT

  両方のファイルに共通の行を整形する。
- --line-format=FORMAT

  全ての行を整形する。上記の全てのオプションを指定した場合に等しい。

行形式では、普通の文字はそれ自身を表す。変換指定は '%' で始まり、以下の形式をとる:

- %l

  行の内容を意味する。行末尾の改行はあっても含まない。この形式では、行に改行があるかどうかは無視される。
- %L

  行の内容を意味する。行末尾の改行があればそれも含む。行に改行がなければ、改行はないままになる。
- %%

‘%’を表す。

- %c’C’’

  ここで C は文字 1 文字で、C を表す。C にバックスラッシュやアポストロフィは指定できない。例えば‘%c’:’’はコロンを表し、これは if-then-else 形式の then 部分でもコロンとして解釈される。通常はコロンは then 部分の終わりとして扱われる。

- %c’¥O’’

  ここで 0 は 1 桁から 3 桁までの 8 進数字であり、8 進のコード 0 に対応する文字を表す。例えば‘%c’¥0’’はナル文字になる。

- Fn

  ここで F は printf(3)の変換指定で、F により整形された行番号を表す。例えば‘%.5dN’は行番号を‘%.5d’という書式で整形して表示する。printf 変換指定の詳細は、上記の行グループ形式のサブセクションを見よ。

デフォルトの行形式は‘%l’に改行文字を続けたものである。入力にタブ文字があり、それが出力行の桁揃えに重要である場合には、‘%l’や‘%L’の行指定をタブストップの直後に置くとよい(すなわち‘%l’や‘%L’の前にタブ文字を置けばよい)。あるいは-t オプションを用いるのもよいだろう。

行形式と行グループ形式を同時に用いると、様々な形式指定が可能となる。例えば、以下のコマンドは diff の通常の形式と似た形式の指定である。これを修正すれば、diff の出力を微調整することが可能になる。

```
diff ¥
   --old-line-format='< %l
' ¥
   --new-line-format='> %l
' ¥
   --old-group-format='%df%(f=l?:,%dl)d%dE
%<' ¥
   --new-group-format='%dea%dF%(F=L?:,%dL)
%>' ¥
   --changed-group-format='%df%(f=l?:,%dl)c%dF%(F=L?:,%dL)
%<---
%>' ¥
   --unchanged-group-format='' ¥
   old new
```

ディレクトリの比較

diff への 2 つの引数がディレクトリだった場合、両方のディレクトリにそれぞれのファイルが、ファイル名のアルファベット順に比較される。通常はファイルのペアに違いが全くなければ、何も出力しない。しかし-s オプションを用いると、同一のファイルも報告する。両方のディレクトリに同名のサブディレクトリがあると、通常 diff は報告だけしてサブディレクトリ以下のファイルは比較しない。しかし-r オプションを用いると、ディレクトリツリーを辿れる限り、対応する全てのファイルを比較する。

片方のディレクトリだけにあるファイルに対しては、diff は通常存在するファイルの内容を表示せず、ファイルが片方にあって他方にはないことだけを報告する。diff の振舞いを変えて、他方のディレクトリにもファイルが空の状態で存在するかのように動作させることもできる。すなわち diff は実際に存在するファイルの内容をすべて出力する。(この出力は、ファイルが第 1 ディレクトリにあれば削除、第 2 ディレクトリにあれば挿入となる。)この指定には-N オプションを使う。

古いほうのディレクトリに大きなファイルがあって、新しいほうにはない場合、-N オプションの代わりに-P オプションを用いるとパッチの大きさを小さくできる。-P オプションは-N オプションと似ているが、第 2 ディレクトリにあるファイルの内容だけを出力に挿入し、第 1 ディレクトリだけにあるファイルは無視する(すなわち、追加されたファイルだけを扱う)。そして、パッチを当てる前に消去されたファイルを削除するよう、パッチの先頭にパッチを当てるユーザーへの指示を書く。

ディレクトリの比較時に特定のファイルを無視させるには、-x PATTERN オプションを用いる。シェルとは異なり、ファイル名の先頭のピリオドは、パターン先頭のワイルドカードにマッチする。シェルによって展開されないよう、PATTERN はクォート記号で囲うべきである。例えば '-x' *.[ao]' ' は '.a' や '.o' で終わる名前のファイルをすべて無視する。このオプションは、複数指定するとそれぞれが有効になる。例えば '-x' RCS' -x' *,v' ' というオプションを指定すると、ファイル名が 'RCS' だったり '*,v' で終わるようなファイルとサブディレクトリをすべて無視する。

返り値

diff は以下の値のどれかで終了する:

- 0
  全く変更がなかった。
- 1
  変更があった。
- 2
  何らかのエラーが起こった。

関連項目

cmp(1)、comm(1)、diff3(1)、ed(1)、patch(1)、pr(1)、sdiff(1)

6.2.5.8 パターンにマッチする行を表示する (grep,、egrep、fgrep、zgrep)

書式

grep [options] PATTERN [FILE...]

grep [options] [-e PATTERN | -f FILE] [FILE...]

説明

grep は、FILE で名前を指定された入力ファイル(ファイルが指定されてないか、file の部分に-が指定された場合は標準入力)を読み込み、与えられた PATTERN にマッチする部分を含む行を探します。デフォルト動作では、grep はマッチした行を表示します。

さらに、2 つのプログラム egrep と fgrep を利用可能です。egrep は grep-E と同じです。fgrep は grep-F と同じです。zgrep は grep-Z と同じです。

オプション

- -A NUM、--after-context=NUM

  NUM で指定した行数だけ、パターンにマッチした行の後の行も表示します。
- -a、--text

  バイナリファイルをテキストファイルであるかのように処理します。これは--binary-files=text オプションと等価です。
- -B NUM、--before-context=NUM

  NUM で指定した行数だけ、パターンにマッチした行の前の行も表示します。
- -C [NUM]、-NUM、--context[=NUM]

  NUM で指定した行数(デフォルトは 2)だけ、パターンにマッチした行の前後の行も表示します。
- -b、--byte-offset

  各出力行の前に、入力ファイルの先頭からのバイト単位のオフセットを表示します。
- --binary-files=TYPE

  ファイルの最初の数バイトが、ファイルの内容がバイナリファイルであることを示す場合、ファイルのタイプを TYPE であると仮定します。デフォルトでは TYPE は binary であり、grep は通常、バイナリファイルの一致を示す一行メッセージを表示するか、マッチしない場合にはなにも表示しません。TYPE が without-match の場合、grep はバイナリファイルはマッチしないものと仮定します。これは-I オプションと等価です。TYPE が text の場合、grep はバイナリファイルをテキストであるかのように扱います。これは-a オプションと等価です。警告:grep --binary-files=text はバイナリのゴミを表示する可能性があります。出力先が端末である場合で、端末ドライバがこのゴミの一部をコマンドであると解釈する場合、このゴミが悪い副作用をおよぼす可能性があります。
- -c、--count

  通常の出力はせず、各入力ファイルについてマッチした行数を表示します。-v、--invert-match オプションと共に指定した場合は、マッチしなかった行数を表示します(下記参照)。

- -d ACTION、--directories=ACTION

  入力ファイルがディレクトリの場合に、ACTION を使ってその処理を行います。デフォルトでは ACTION は read であり、ディレクトリを普通のファイルであるかの様に読み取る事を意味します。ACTION が skip なら、ディレクトリを黙って読み飛ばします。ACTION が recurse なら、grep は各ディレクトリ下のすべてのファイルを再帰的に読み取ります。これは-r オプションと等価です。

- -E、--extended-regexp

  PATTERN を拡張された正規表現として扱います(下記参照)。

- -e PATTERN、--regexp=PATTERN

  PATTERN をパターンとして指定します。-で始まるパターンを保護するために有効です。

- -F、--fixed-strings

  PATTERN を改行で区切られた固定文字列のリストとして扱います。その文字列のどれかとマッチするかを調べます。

- -f FILE、--file=FILE

  パターンを FILE から 1 行ごとに読み込みます。空のファイルはパターンを含まないので、何にもマッチしません。

- -G、--basic-regexp

  PATTERN を基本的な正規表現として扱います(下記参照)。デフォルトです。

- -H、--with-filename

  各々のマッチに対してファイル名を表示します。

- -h、--no-filename

  複数ファイルを検索した時に、出力の前にファイル名を付けることを抑制します。

- --help

  簡単なヘルプメッセージを出力します。

- -I

  バイナリファイルをマッチするデータを含まないかのように処理します。これは --binary-files=without-match オプションと等価です。

- -i、--ignore-case

  PATTERN と入力ファイルの双方で、英大文字と小文字の区別をしないようにします。

- -L、--files-without-match

  通常の出力はせず、このオプションを指定しなかったときに全く出力されない入力ファイルの名前を表示します。スキャン動作は最初のマッチで終了します。

- -l、--files-with-matches

  通常の出力はせず、このオプションを指定しなかったときに出力される入力ファイルの名前を表示します。スキャン動作は最初のマッチで終了します。

- --mmap

可能ならば、デフォルトの read(2)システムコールの代わりに mmap(2)システムコールを使って入力を読み取ります。ある状況において、--mmap はよりよい性能をもたらします。しかし、grep の動作中に入力ファイルが小さくなるか、または I/O エラーが生じた場合に、--mmap は(コアダンプを含む)未定義の動作を引き起こす可能性があります。

- -n、--line-number

  各出力行の前に、入力ファイルにおける行番号を表示します。

- -q、--quiet、--silent

  沈黙。通常の出力を抑止します。スキャン動作は最初のマッチで終了します。下記の-s や --no-messages オプションも参照。

- -r、--recursive

  各ディレクトリ下のすべてのファイルを再帰的に読み取ります。これは-d recurse オプションと等価です。

- -s、--no-messages

  指定されたファイルが存在しないことや読み込みできないことを示すエラーメッセージを抑止します。移植性に関する注:GNU grep とは異なり、伝統的な grep は POSIX.2 に適合していませんでした。なぜなら、伝統的な grep には-q オプションが無く、-s オプションは GNU grep の-q オプションの様に動作したからです。伝統的な grep へ移植可能であることを意図したシェルスクリプトは、-q と-s を両方とも使わずに、出力を/dev/null へリダイレクトすべきです。

- -U、--binary

  ファイルをバイナリとして扱います。デフォルトでは、MS-DOS と MS Windows 環境下で grep は、ファイルから読み取った最初の 32KB の内容を見て、ファイルタイプを推測します。grep はファイルをテキストファイルと判断した場合、オリジナルのファイル内容から(^と$が使われている正規表現を正しく動作させるために)CR 文字を取り除きます。-U を指定すると、この当て推量を抑制し、すべてのファイルを読み取ってマッチ機構へそのまま渡します。もしファイルが各行の末尾に CR/LF の組みを持つテキストファイルなら、このオプションは正規表現を役に立たなくさせることがあるでしょう。このオプションは MS-DOS と MS-Windows 以外のプラットフォームでは効果がありません。

- -u、--unix-byte-offsets

  unix 形式のバイト単位オフセットを報告します。このスイッチを指定すると grep は、ファイルが unix 形式のテキストファイル、すなわち、CR 文字が取り除かれたファイルであるかのごとくにバイト単位オフセットを報告します。このことは grep を Unix マシンで動作させたのと同じ結果を生成します。このオプションは-b オプションも使用しない限り効果がありません。MS-DOS と MS-Windows 以外のプラットフォームでは効果がありません。

- -V、--version

  grep のバージョン番号を標準エラー出力に表示します。バグレポートには、この番号を付記してください(下記参照)。

- -v、--invert-match

  結果を反転し、マッチしなかった行を選択します。

- -w、--word-regexp

  完全な語にマッチする行のみを選択します。マッチする部分文字列が行頭から始まっているか、単語構成文字以外の文字が前にあることがテストされます。同様に、マッチする部分文字列が行末まであるか、単語構成文字以外の文字が後にある必要があります。単語構成文字とは、レター・数字・アンダスコアです。

- -x、--line-regexp

  行全体と正確にマッチする行のみを選択します。

- -y

  -i と同じ意味を持つ旧式のオプションです。

- --null

  通常ファイル名の後に続く文字の代わりにバイト 0(ASCII NUL 文字)を出力します。例えば、grep -l --null は各ファイル名の後に、通常の newline ではなくバイト 0 を出力します。このオプションを指定すると、newline 等の例外的な文字を含むファイル名に直面した場合でも出力が明白になります。このオプションを find -print0、perl -0、sort -z、xargs -0 等のコマンドと共に使用すれば、任意のファイル名を処理できます。ファイル名が newline 文字を含んでいても処理可能です。

- -Z、--decompress

  検索を開始する前に入力データを伸長します。このオプションは zlib ライブラリと共にコンパイルした場合のみ使用可能です。

正規表現

正規表現は、文字列の集合を表現するパターンの事です。数式表現と同様に、より小さな表現を組み合わせるさまざまな演算子を用いる事で、正規表現を組み立てます。

grep は、「基本」正規表現と「拡張」正規表現の 2 種類の正規表現文法を扱う事ができます。GNU grep では、どちらの正規表現文法も機能的な違いはありません。他の実装では、基本正規表現は拡張正規表現より能力が低くなっています。ここでは、拡張正規表現について説明します。基本正規表現との違いは、その後に説明します。

正規表現の基本単位は、1 文字にマッチする正規表現です。レターと数字を含む多くの文字は、それ自身にマッチする正規表現です。また、特殊な意味を持つメタ文字も、その文字の前にバックスラッシュを付けると、その本来の文字にマッチするようになります。

[と]で囲まれた文字のリストは、そのリスト中に含まれるどれか 1 文字にマッチします。ただし、リストの先頭がキャレット^の場合は、そのリストに含まれない文字にマッチします。例えば、正規表現[0123456789]

は数字 1 文字にマッチします。文字の範囲は最初と最後の文字をハイフン('-')でつなぐことで指定できます。最後に、特定の名前を持つ文字クラスがあらかじめ定義されています。名前が内容を示しており、それらは、[:alnum:]、 [:alpha:]、[:cntrl:]、[:digit:]、[:graph:]、[:lower:]、[:print:]、 [:punct:]、[:space:]、[:upper:]、[:xdigit:]です。例えば、[[:alnum:]]は[0-9A-Za-z]と同じですが、後者は POSIX ロケールや ASCII コード順に依存しますので、前者の方がロケールや文字集合に依存しません。(クラス名の中の角括弧はシンボル名の一部であり、リストを区切る角括弧とは別に指定する必要があることに注意)リストの中では、ほとんどのメタ文字は通常の文字として扱われます。リテラル]を含めるには、この文字をリストの先頭に置いてください。同様に、リテラル^を含めるには、この文字をリストの先頭以外に置いてください。リテラル-を含めるには、この文字をリストの最後に置いてください。

ピリオド.は、任意の 1 文字にマッチします。シンボル¥w は [[:alnum:]]と同じ意味で、シンボル¥W は [^[:alnum:]]と同じ意味です。

キャレット^と、ドル記号$は、それぞれ行頭と行末の空文字列にマッチするメタ文字です。シンボル¥<とシンボル¥>は、それぞれ単語の先頭と末尾の空文字列にマッチするメタ文字です。シンボル¥b は単語の端の空文字列にマッチします。シンボル¥B は単語の端以外の空文字列にマッチします。

正規表現の後には、繰り返し演算子のどれかが続くことがあります。

- ?
  直前の項目はオプションであり、最大 1 回マッチします。
- *
  直前の項目は 0 回以上マッチします。
- +
  直前の項目は 1 回以上マッチします。
- {n}
  直前の項目は厳密に n 回マッチします。
- {n,}
  直前の項目は n 回以上マッチします。
- {n,m}
  直前の項目は、最低 n 回、最大 m 回マッチします。

2 つの正規表現は結合可能です。結果としてできあがる正規表現は、結合された 2 つの部分表現にそれぞれマッチする 2 つの部分文字列を結合した任意の文字列にマッチします。

2 つの正規表現は中置き型演算子|で繋ぐことが可能です。結果としてできあがる正規表現は、どちらかの部分表現にマッチする任意の文字列にマッチします。

繰り返しは結合に優先します。また結合は選択に優先します。これらの優先規則を無効とするために、部分表現全体を括弧で囲むことが可能です。

n が 1 つの数字であるような後方参照¥n は、正規表現中の括弧で囲まれた n 番目の部分表現がマッチした文字列とマッチします。

基本正規表現では、メタ文字?、+、{、|、(、)は、その特殊な意味を失います。代わりに、バックスラッシュを付けた¥?、¥+、¥{、¥|、¥(、¥)を使用してください。

伝統的な egrep は、メタ文字{をサポートしませんでした。また、このメタ文字の代わりに¥{をサポートする egrep 実装もいくつか存在するので、移植可能なスクリプトでは、リテラル{にマッチさせるために egrep パターンで{を使うことは避けて[{]を使うべきです。

GNU egrep は、{が不正な範囲指定の始まりであるなら特殊文字ではない、と想定して、伝統的な使用法のサポートを試みます。例えば、シェルコマンド egrep '{1' は正規表現の文法エラーを報告せずに、2 文字の文字列{1 を検索します。POSIX.2 は、この動作を一つの拡張として許可していますが、移植可能なスクリプトではこの使用法を避けるべきです。

環境変数

- GREP_OPTIONS

  この変数は明示的なオプションの前に指定されるデフォルトオプションを指定します。例えば、もし GREP_OPTIONS が' --binary-files=without-match --directories=skip' である場合、grep は 2 つのオプション--binary-files=without-match と--directo-ries=skip が明示的なオプションの前に指定されている様に動作します。オプションの指定は空白によって区切られます。バックスラッシュは次の文字をエスケープします。これは空白やバックスラッシュを含むオプションを指定するために用いられます。
- LC_ALL、LC_MESSAGES、LANG

  これらの変数は grep がメッセージに使用する言語を決定する LC_MESSAGES を指定します。ロケールはこれらの変数のうち最初に設定されているものにより決定されます。もしこれらの変数全てが設定されていない場合、またはメッセージカタログがインストールされていない場合、または grep が国際言語サポートつき(NLS)でコンパイルされてない場合には、アメリカンイングリッシュが用いられます。
- LC_ALL、LC_CTYPE、LANG

  これらの変数は、例えばどの文字が空白であるかなど、文字の種類を決定する LC_CTYPE を指定します。ロケールはこれらの変数のうち最初に設定されているものにより決定されます。もしこれらの変数全てが設定されていない場合、またはメッセージカタログがインストールされていない場合、または grep が国際言語サポートつき(NLS)でコンパイルされてない場合には、POSIX ロケールが用いられます。
- POSIXLY_CORRECT

  設定されている場合、grep は POSIX.2 として動作し、それ以外の場合は grep は他の GNU プ

ログラムのように動作します。POSIX.2 ではファイル名の後に続くオプションはファイル名として扱われます。デフォルトでは、このようなオプションはオペランドリストの先頭に並び変えられて、オプションとして扱われます。また、POSIX.2 では認識できないオプションは“不法(illegal)”であると診断されますが、法律に違反しているわけではないので、デフォルトではこれらは“不正(invalid)”であると診断されます。

返り値

通常、パターンにマッチした行が見つかった場合は0を、見つからなかった場合は1を返します。(ただし、-v オプションを指定した場合は、逆になります)。パターンに文法エラーが存在したり、入力ファイルにアクセスできないなどのシステムエラーが発生した場合は、2 を返します。

6.2.5.9 タブをスペースに変換する (expand)

書式

expand [-i] [-TAB1[,TAB2...] [-t TAB1[,TAB2...] [--initial] [--tabs=TAB1[,TAB2...] [FILE...]

expand [--help] [--version]

説明

expand は指定された FILE の内容を標準出力に書き込む。その際、タブ文字を適当な数のスペースに置き換える。FILE が 1 つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

デフォルトでは expand はすべてのタブをスペースに変換する。バックスペース文字は出力に保存され、タブの計算における桁数を減らす作用をもたらす。デフォルトの動作は '-t 8' が指定された場合と同じ(タブを 8 桁おきに設定する)。

オプション

- -TAB1[,TAB2...]、-t TAB1[,TAB2...]、--tabs=TAB1[,TAB2...]
  タブストップが 1 つだけ与えられた場合には、タブを TAB1 おきに設定する(デフォルトは 8)。その他の場合は、タブを TAB1 桁、TAB2 桁...に設定し(桁数は 0 から数え始める)、与えられた数以降のタブはスペース文字 1 つに置き換える。タブストップを--tabs オプションで指定する際には、それぞれの間は空白またはコンマで区切る。短縮形式-TAB...は推奨されない。
- -i、--initial
  それぞれの行の先頭タブ(initial tabs)のみをスペースに変換する。「先頭タブ」とは、行先頭か

ら続くタブ文字の集まりのこと。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.10 テキストを段落に整形する (fmt)

書式

```
fmt [-ctsu] [-WIDTH] [-p REFIX] [-w WIDTH] [--crown-margin] [--split-only]
[--tagged-paragraph]
[--uniform-spacing] [--prefix=PREFIX] [--width=WIDTH] [FILE...]

fmt [--help] [--version]
```

説明

fmt は行を結合したり隙間を埋めたりして、出力各行を与えられた文字数に揃える(正確には越えないようにする)。デフォルトは 75 桁。fmt は FILE 引数(ファイルが一つも指定されなかった場合は標準入力)から読み、標準出力に書く。

デフォルトでは、空行・単語間のスペースの数・インデントは出力に保存される。インデントの異なる連続行は結合しない。タブは入力で一度スペースに変換され、出力で挿入される。

fmt は文の最後で改行しようとする。また文の最初の単語の直後や、文の最後の単語の直前での改行は避けようとする。「文の区切り」は、段落の最後か、'.?!' のどれかで終わる単語に 2 つのスペースまたは続いたものとして定義される。ただし括弧や引用符の内部では文区切りとはみなされない。TeX(1)と同じく、fmt も「段落」全体を読み込んでから改行位置を決める。このアルゴリズムは、"Breaking Paragraphs into Lines" (Donald E. Knuth and Michael F. Plass, Software--Practice and Experience', 11 (1981) 1119-1184) で提案されたものをちょっと変更したものである。

オプション

- -c、--crown-margin

  クラウンマージンモード(crown margin mode)。段落の最初の二行のインデントを保存し、引き続く行の左マージンをそれぞれ第二行のものに揃える。

- -p PREFIX、--prefix=PREFIX

  PREFIX で始まる行(スペースが前置されていても良い)だけを再配置する。PREFIX(と前置されるスペース)は整形前にいったん削除され、整形された出力行に再び挿入される。利用法と

しては、例えばある種のプログラムのコメントだけを整形して、コード部分は変更したくない場合などが考えられる。

- -s、--split-only

  行分割だけを行う。短い行を結合して長い行を生成する作業を行わない。プログラムコードなどの「整形済み」テキストが不正に結合されるのを避けることができる。

- -t、--tagged-margin

  タグ付き段落モード(tagged paragraph mode)。クラウンモードと似ているが、段落の最初の行のインデントは第二行のものとは異なっていなければならない。同じである場合は、最初の行は一行だけからなる段落とみなされる。

- -u、--uniform-spacing

  一様なスペース配置(uniform spacing)。単語間のスペースをスペース文字ひとつに減らす。ただし文の間には二つのスペース文字が用いられる。

- -WIDTH、-w WIDTH、--width=WIDTH

  出力行を WIDTH 桁まで埋めようとする(デフォルトは 75)。fmt は各行の長さを揃えるための空間を確保する目的から、およそ 7%ほど行を短くする傾向にある。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.11 入力行を指定された幅にあわせて折りたたむ

書式

```
fold [-bs] [-w WIDTH] [--bytes] [--spaces] [--width=WIDTH] [FILE...]
fold [--help] [--version]
```

説明

fold は指定されたファイルの内容を標準出力に書き込む。その際、長い行は複数の短い行に分割する。FILE が一つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

デフォルトでは、fold は 80 桁以上の行を分割する。出力は必要な分の行数になる。

デフォルトでは桁数の評価は画面出力のイメージをもとに行われる。したがってタブ文字は複数桁に数えられることが多いだろうし、バックスペース文字は桁数を減らすことになり、復帰(carriagereturn)文字は桁数のカウントを 0 に戻すことになる。

オプション

- -b、--bytes

  桁数ではなくバイト数をカウントする。したがってタブ、バックスペース、復帰の各文字は、他の文字と同じくカウントをひとつだけ増す効果のみを持つことになる。

- -s、--spaces

  単語(word)の境界で行の分割を行う。行に空白(blank)があれば、行の最大長を越えないうち、最も大きな桁にある空白で改行が行われる。行に空白がひとつもなければ、行の最大長の位置で改行が行われる(通常の動作と同じ)。

- -w WIDTH、--width=WIDTH

  行の最大長を WIDTH 桁にする(デフォルトは 80)。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.12 ファイルの最初の部分を表示する (head)

書式

```
head [-LINES] [-qv] [-c BYTES] [-n LINES] [--bytes=BYTES]
[--lines=LINES] [--quiet] [--silent] [--verbose] [FILE...]

head [--help] [--version]
```

説明

head は引数に指定された FILE の最初の部分(デフォルトは 10 行)を表示する。FILE が 1 つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

複数の FILE が指定されたときは, head はそれぞれの前に、以下の内容の 1 行のヘッダを各ファイルの前に出力する:

==> FILENAME <==

head は 2 つのオプション形式を受付ける。新しい形式は、数値をオプションの引数として与える('-q -n 1')ものであり、古い形式はオプション文字の前に数値を指定する('-1q')ものである。

オプション

- -LINES

  このオプションは最初に指定されたときに限って認識される。LINES は 10 進数の数値。単位を表す文字(‘b’、‘k’、‘m’)を後置したり(それぞれの意味は‘-c’のものと同じ)、行単位のカウントを指定する ‘l’を後置したり、他のオプション文字(‘cqv’)を後置したりできる。何も文字が後置されなかった場合は‘l’が指定されたのと同じことになる。このオプションは推奨されない。代わりに--lines を使うこと。
- -c BYTES、--bytes=BYTES

  行単位ではなく、先頭の BYTES バイトを表示する。‘b’を追加すると BYTES の 512 倍、‘k’は 1024 倍、‘m’は 1048576 倍を指定したことになる。
- -n LINES、--lines=LINES

  最初の LINES 行を表示する。
- -q、--quiet、--silent

  ファイル名のヘッダを出力しない。
- -v、--verbose

  常にファイル名のヘッダを出力する。
- --help

  標準出力に使用方法のメッセージを出力して正常終了する。
- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.13 ファイルをコピーし、その属性を設定する (install)

書式

```
install [options] [-s] [--strip] source dest
install [options] [-s] [--strip] source... directory
install [options] [-d,--directory] directory...
```

オプション (簡略形式):

```
[-bcpvD] [-g group] [-m mode] [-o owner] [-S suffix] [-V {numbered,existing,simple}]
[--preserve-timestamps] [--target-directory=dir] [--help] [--version] [--]
```

説明

install はファイルをコピーし、そのアクセス権を設定する。可能ならば所有者とグループを設定する。

1 番目の呼び出し形式では、ファイル source が目的のファイル dest にコピーされる。2 番目の形式では、複数のファイル source のそれぞれが指定ディレクトリ directory にコピーされる。最後の形式では、個々のディレクトリ directory が(親ディレクトリがない場合はそれも含めて)作成される。

install は cp と似ているが、コピー先ファイルの属性を制御できる。プログラムを指定ディレクトリにコピーするために、Makefile 内でよく用いられる。ファイルをそれ自身へコピーすることはできない。

オプション

- -c

  無視される。古い Uxix 版の install との互換性のためにある。

- -d、--directory

  指定されたディレクトリを作成する。親ディレクトリが存在しない場合は、それも作成する。所有者・グループ・モードを、コマンドラインで指定された設定、またはデフォルトの設定にする。このとき作成した親ディレクトリにも同じ属性を与える。

- -g group、--group=group

  インストールされたファイルやディレクトリの所有グループを group に設定する。デフォルトではプロセスが属する現在のグループになる。group はグループ名でも数字のグループ ID でもよい。

- -m mode、--mode=mode

  インストールされたファイルやディレクトリのアクセス権を mode に設定する。モードは 0 を基点とし、8 進数、chmod におけるシンボルモードのどちらでもよい。デフォルトのモードは 0755 で、所有者の読み取り・書き込み・実行、グループとその他の人の読み取り・実行が可能である。

- -o owner、--owner=owner

  install が適正な権利を持っている(root で実行された)場合、インストールされたファイルやディレクトリの所有者を owner に設定する。デフォルトは 'root' である。owner はユーザー名でも数字のユーザーID でもよい。

- -p、--preserve-timestamps

  インストールされたファイルの最終アクセス時刻と最終修正時刻をそれぞれコピー元ファイルと合わせる。

  このオプションなしでファイルがインストールされると、最終アクセス時刻と最終修正時刻は共にインストール時刻に合わせられる。このオプションは、インストールされたファイルの最終修正時刻を、「いつインストールされたか」ではなく「いつ作られたか」を記憶させるために使いたい場合に便利である。

- -s、--strip

  インストールされたバイナリ実行ファイルからシンボルテーブルを strip する。

- --target-directory=dir

  コマンドラインの最後の引き数ではなく、オプションでインストール先ディレクトリを指定する。xargs(1)と一緒に使う場合に役立つ。

- -D

  dest にコピーするために必要な全ディレクトリ構造を(それがない場合は)前もって作成する。

それから source を dest にコピーする。1 番目の形式の場合に便利である。

- -v、--verbose

  インストールする前にそれぞれのファイル名を出力する。

GNU バックアップオプション

GNU 版のプログラム cp、mv、ln、install、patch は、上書き・修正・削除といった場合に、指示すればファイルのバックアップを作成する。バックアップファイルを必要とする場合は-b オプションで指定する。どのような名前にするかは--backup オプションで指定する。バックアップファイルの名前を、ファイル名に拡張子を追加する形で与えるようにしたい場合、この拡張子を-S オプションで指示する。

- -b、--backup[=method]

  上書きもしくは削除の必要がある場合には、ファイルのバックアップを作成する。-b が引き数をとらない点に注意すること。

- -S suffix、--suffix=suffix

  SUFFIX をバックアップファイルそれぞれに付け加える。このオプションが指定されていない場合、環境変数 SIMPLE_BACKUP_SUFFIX に設定されている値が使われる。SIMPLE_BACKUP_SUFFIX が設定されていない場合のデフォルトは '~' である。

- -V method、--version-control=method

  バックアップファイルの命名方法を指定する。引き数 method には、'numbered'(または 't')、'existing'(または 'nil')、'never' (または 'simple')を指定できる。このオプションが指定されていない場合、環境変数 VERSION_CONTROL の値が使われる。 VERSION_CONTROL が設定されていない場合のデフォルトは 'existing' である。

  このオプションは Emacs 変数の 'version-control' に対応している。有効な method は以下の通り。(他と重複しない短縮形が使える):

  t、numbered 常に番号の拡張子を持つバックアップが作られる。

  nil、existing 番号の拡張子を持つバックアップがすでにある場合には番号の拡張子を持つバックアップを、そうでない場合には単純なバックアップを作成する。

  never、simple 常に単純なバックアップが作られる。

  このオプションは推奨されない。代りに--backup=method を使うこと。

GNU 標準オプション

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

- --

オプションリストの終りを示す。

環境変数

変数 LANG、LC_ALL、LC_CTYPE、LC_MESSAGES が通常の意味を持つ。GNU 版では、変数 SIMPLE_BACKUP_SUFFIX と VERSION_CONTROL がバックアップファイルの命名法を上で説明した方法で管理する。

準拠

BSD 4.2(-c、-m、-o、-g、-s オプションがある)。

6.2.5.14 二つのファイルを読み、フィールドが共通な行を結合する (join)

書式

```
join [-i] [-1 FIELD] [-2 FIELD] [-a FILE-NUMBER] [-e STRING] [-j FIELD] [-j1 FIELD] [-j2 FIELD]
[-o FIELD-LIST] [-t CHAR] [-v FILE-NUMBER] [--ignore-case] FILE1 [FILE2]

join [--help] [--version]
```

説明

join は join フィールドが一致している入力行の各ペアを標準出力に表示する。一方の FILE が与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

FILE1 と FILE2 は実行前に join フィールドの昇順にソートしておかなければならない(数値順のソートはだめ)。-t オプションが与えられなかった場合は、ファイルは join フィールドの先頭にある空白を無視してソートしておかなければならない('sort -b' のようなかたち)。--ignore-case オプションを与える場合には、各行は join フィールドの英大文字小文字を無視してソートしておかなければならない('sort -f' のようなかたち)。

デフォルトの動作は以下の通り:join フィールドは各行の先頭に来る。入力は 1 つ以上の空白でフィールドに分割され、行頭の空白は無視される。出力のフィールドは 1 つのスペースで区切られる。出力行は、join フィールド、FILE1 の残りのフィールド、FILE2 の残りのフィールドからなる。

オプション

- -1 FIELD、-j1 FIELD

FILE1 の FIELD 番目のフィールドを用いて join を行う(FIELD は正の整数)。

- -2 FIELD、-j2 FIELD

  FILE2 の FIELD 番目のフィールドを用いて join を行う(FIELD は正の整数)。

- -a FILE-LIST

  FILE-NUMBER('1' または '2')のファイルにあるペアにならなかった行を、通常の出力に追加して表示する。

- -e STRING

  入力にフィールドがなかった場合、それに対応する出力フィールドを STRING にする。

- -i、--ignore-case

  キーを比較するときに英大文字小文字の違いを無視する。このオプションを指定するときは、入力ファイルも同じように整列させておかなければならない。このような整列を行うには 'sort -f' を使えば良い。

- -j FIELD

  -1 FIELD -2 FIELD と同じ。

- -o FIELD-LIST...

  出力行のフォーマットに FIELD-LIST を用いる。FIELD-LIST の各要素は、一文字 '0' または M.N の形式である。ここで M はファイル番号で '1' または '2' である。N は正の整数で、フィールドの番号である。

  フィールド指定 '0' は join フィールドを表す。ほとんどの場合は、 '0' の機能は M.N を用いて join フィールドを明示的に指定するやり方でも再現可能であろう。しかし、(-a や-v オプションを使ったときに)ペアにならなかった行も表示する場合は、両方のファイルにそのような行があると、FIELD-LIST で M.N を使うやり方では join フィールドを指定することはできない。join にこの機能を与えるため、POSIX で '0' フィールド指定の記述が発明された。

  FIELD-LIST の各要素はコンマまたは空白で区切られる。一つの-o オプションの後に複数の FIELD-LIST 引数を指定することも出来る。-o オプション以降の全てのリストの値は結合される。FIELD-LIST の指定は、(-a や-v オプションに由来するものも含め)全ての出力行に適用される。

- -t CHAR

  入力・出力のフィールド区切りに文字 CHAR を用いる。

- -v FILE-NUMBER

  通常の出力ではなく、FILE-NUMBER('1' または '2')のファイルにある、ペアにならなかった行だけを表示する。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.15 ページャー (less)

書式

```
less -?
less --help
less -V
less --version
less [-[+]aBcCdeEfgGilmMnNqQrsSuUVWwXZ] [-b bufs] [-h lines] [-j line] [-k keyfile]
 [-K character set] [-{oO} logfile] [-p pattern] [-P prompt] [-t tag]
 [-T tagsfile] [-x tab] [-y lines] [-[z] lines] [+[+]cmd] [--] [filename]...
```

(長いオプション名を使う別の書式については、オプションのセクションを参照すること。)

説明

less は more(1)と同様なプログラムであるが、ファイル内での前方移動と同様に後方移動も可能となっている。また、less は起動時に入力ファイル全体を読み込む必要がないため、大きな入力ファイルの場合には vi(1)のようなテキストエディタより起動が速い。less は termcap(システムによっては terminfo)を使用するため、多くの端末で実行できる。ハードコピー端末では、まだサポートが限定されている(ハードコピー端末では、画面の一番上に表示されるべき行にキャレット(^)が付く)。

コマンドは more と vi の両方を基本にしている。(以下の説明において N で表す)10 進数の後に続けて使用できるコマンドもある。

コマンド

以降の説明で^X は control-X を意味する。ESC は ESCAPE キーである。例えば、ESC-v は"ESCAPE"を押した後に"v"を押すという意味である。

- h、H

  ヘルプ。コマンドの概要を表示する。もし、他のコマンドを忘れた時は、このコマンドを思い出すこと。
- SPACE、^V、f、^F

  前方に 1 ウインドウ分(N 行)進む(-z オプションを参照)。行数 N が画面サイズより大きい場合は、最後の行から画面に入りきる分だけを表示する。警告:^V が特殊なリテラル化文字として使われているシステムもある。
- z

  SPACE と似ているが、行数 N を指定すると N が新しいウインドウのサイズになる。
- ESC-SPACE

  SPACE と似ているが、ファイルの終端に達した場合でも画面いっぱいにスクロールする。
- RETURN、^N、e、^E、j、^J

前方に 1 行(N 行)進む。N が画面サイズより大きい場合でも、N 行全てを表示する。

- d、^D

  前方に半画面(N 行)進む。行数 N を指定すると、その後の d コマンドと u コマンドの新しいデフォルトサイズになる。

- b、^B、ESC-v

  後方に 1 ウインドウ分(N 行)戻る(-z オプションを参照)。N が画面サイズより大きい場合は、最後の行から画面に入りきる分だけを表示する。

- w

  ESC-v と似ているが、N を指定すると、N が新しいウインドウのサイズになる。

- y、^Y、^P、k、^K

  後方に 1 行(N 行)戻る。N が画面サイズより大きい場合でも、N 行全てを表示する。警告:^Y が特殊なジョブ制御文字として使われているシステムもある。

- u、^U

  後方に半画面(N 行)戻る。N を指定すると、その後の d コマンドと u コマンドの新しいデフォルトサイズになる。

- ESC-)、RIGHTARROW

  画面を右方向に画面幅の半分(N 文字分)スクロールする(-#オプションを参照)。テキストがスクロールしている間は、(行を切り取る) -S オプションが有効になっているように動作する。文字数 N を入力する場合は、右矢印キーが行編集コマンド(行編集のセクションを参照)で使われるため、ESC-)を使用しなければならない。

- ESC-(、LEFTARROW

  画面を左方向に画面幅の半分(N 文字分)スクロールする(-#オプションを参照)。

- r、^R、^L

  画面を再描画する。

- R

  画面を再描画し、バッファされた入力は破棄する。ファイルを閲覧中に、ファイルが変更された場合に有効である。

- F

  前方にスクロールし、ファイルの終端に達しても読み続ける。通常このコマンドは、既にファイルの終端に達している場合に使われる。閲覧中に大きくなり続けるファイルの末尾をモニターすることができる (″tail-f″コマンドと同じような動作をする)。

- g、<、ESC-<

  ファイルの第 1 行目(第 N 行目)に移動する(警告:N が大きいと遅くなる)。

- G、>、ESC->

  ファイルの最終行(第 N 行目)に移動する(警告:N が大きい場合や、N が指定されない場合、また、ファイルではなく標準入力を読み込んでいる場合に遅くなる)。

- p、%

  ファイルの N パーセント目の位置に移動する。N は 0 から 100 の間でなければならない。

- {

  画面に表示されている先頭行に左中括弧がある場合、{コマンドを使うと対応する右中括弧の位置に移動する。対応する右中括弧は画面の最終行に表示される。2 つ以上の左中括弧が画面の先頭行にある場合、数字 N は第 N 個目の括弧を指定するために使われる。

- }

  画面に表示されている最終行に右中括弧がある場合、}コマンドを使うと対応する左中括弧の位置に移動する。対応した左中括弧は画面の先頭行に表示される。2 つ以上の右中括弧が画面の最終行にある場合、数字 N は第 N 個目の括弧を指定するために使われる。

- (

  {コマンドと似ているが、中括弧ではなく小括弧に適用される。

-)

  }コマンドと似ているが、中括弧ではなく小括弧に適用される。

- [

  {コマンドと似ているが、中括弧ではなく大括弧に適用される。

-]

  }コマンドと似ているが、中括弧ではなく大括弧に適用される。

- ESC-^F

  2 つの文字を続けて入力することで、{コマンドと同様の動作をする。2 つの文字はそれぞれ開括弧と閉括弧として使われる。例えば、"ESC ^F < >"を実行すると、画面の先頭行に表示されている<に対応する>に進むことができる。

- ESC-^B

  2 つの文字を続けて入力することで、}コマンドと同様の動作をする。2 つの文字はそれぞれ開括弧と閉括弧として使われる。例えば、"ESC ^B < >"を実行すると、画面の最終行に表示されている>に対応する<に戻ることができる。

- m

  任意の小文字を続けて入力することで、現在の位置を入力した文字でマークする。

- ’（シングルクォート）

  任意の小文字を続けて入力することで、以前この文字でマークした位置に戻る。もう 1 つのシングルクォートを続けることで、シングルクォートコマンドで「大きく」移動する前の位置に戻る。^, $で、それぞれファイルの先頭行と最終行にジャンプする。新しいファイルを読み込む場合も前のファイルに付けたマークは保存されるので、’コマンドは入力ファイルの切替えに使うことができる。

- ^X^X

  シングルクォートと同じ。

- /pattern

  前方にあるパターンにマッチする 1 番目(N 番目)の行を検索する。このパターンは ed で認識される正規表現である。検索は、画面に表示されている第 2 行から始まる(変更する場合は、-a と-j オプションを参照すること)。

  パターンの前に置く文字のうち、いくつかは特殊文字となる。これはパターンの一部としてではなく、検索方法を変更するために使われる。

- ^N、!

  パターンにマッチしない行を検索する。

- ^E、*

  複数のファイルを検索する。つまり、検索がマッチしないまま現在のファイルの終りに達した場合、コマンドラインのファイルリストにある次のファイルに検索を続行する。

- ^F、@

  現在の画面表示や、-a または-j オプションの設定に関係なく、コマンドラインのファイルリストにある最初のファイルの先頭行から検索を開始する。

- ^K

  現在の画面中でマッチする全てのパターンをハイライト表示する。その際、一番始めにマッチする位置へは移動しない(現在の位置を保持する)。

- ^R

  正規表現のためのメタキャラクタを解釈しない。つまり、単純な文字列比較を行う。

- ?pattern

  後方にあるパターンにマッチする 1 番目(N 番目)の行を検索する。検索は、画面の一番上に表示されている行の前の行から行う。

- /

  コマンドの時と同様に、一部の文字は特殊な役割をもつ。

- ESC-/pattern

  "/*"と同じ。

- ESC-?pattern

  "?*"と同じ。

- n

  前回の検索を繰り返す。すなわち、前回使用した検索パターンにマッチする N 番目の行の検索する。前の検索が^N で変更されている場合、パターンにマッチしない N 番目の行を検索する。前の検索が^E で変更されている場合、現在のファイルに条件を満たす行がないときは次の(あるいは前の)ファイルで検索を続ける。前の検索が^R で変更されている場合、正規表現を用いずに検索を行う。前の検索が^F または^K で変更されている場合、何の変化も起きない。

- N

前回の検索を逆方向に行う。

- ESC-n

  複数のファイルがあるとき、ファイルを越えて直前の検索を繰り返す。前の検索が、*によって変更された場合と同じ。

- ESC-N

  複数のファイルがあるとき、ファイルを越えて逆方向に直前の検索を繰り返す。

- ESC-u

  検索結果のハイライトを元の状態に戻す。現在の検索パターンにマッチした文字列のハイライトをオフにする。もし既に、前の ESC-u コマンドによりハイライトがオフになっている場合は、検索コマンドの実行によってもハイライトはオンに戻る。(-G オプションでもハイライトをオフに切替えることができる。この場合、検索コマンドでハイライトがオンにならない)。

- :e [filename]

  新しいファイルを読み込む。ファイル名の指定がない時は、コマンドラインのファイルリストから「現在の」ファイル(:n と:p コマンドを参照)を再度読み込む。ファイル名の中のパーセント記号(%)は現在のファイル名で置き換えられる。ファイル名の中のシャープ記号 (#)は前に読み込んだファイル名で置き換えられる。しかし、2 つ続けたパーセント記号は、単純に 1 つのパーセント記号に置き換えられる。これは、パーセント記号を含むファイル名を入力できるようにするためである。同様に、2 つ続けたシャープ記号は、単純に 1 つのシャープ記号に置き換えられる。このファイル名は、その後に続く:n コマンドと:p コマンドで閲覧できるように、コマンドラインのファイルリストに挿入される。複数のファイル名を指定した場合は、全てをファイルのリストに加え、最初のファイルを閲覧する。ファイル名が、1 個以上のスペースを含む場合、ファイル名全体がダブルクォートで囲まれていなければならない(″-″オプションを参照)。

- ^X^V、E

  :e コマンドと同じ。警告:^V が特殊なリテラル化文字として使われているシステムもある。そのようなシステムでは、^V を使うことはできない。

- :n

  (コマンドラインで与えられたファイルリストから)次のファイルを読み込む。数字 N が指定されている時は、次の N 番目のファイルを読み込む。

- :p

  コマンドラインのファイルリストにある 1 つ前のファイルを読み込む。数字 N が指定されている時は、前の N 番目のファイルを読み込む。

- :x

  コマンドラインのファイルリストにある一番最初のファイルを読み込む。数字 N が指定されている時は、第 N 番目のファイルを読み込む。

- :d

  現在のファイルをファイルリストから取り除く。

- =、^G、:f

  閲覧中のファイルについて、ファイル名・表示中の最終行の行番号・バイトオフセットといった情報を表示する。可能な場合には、ファイルの長さ・ファイルの行数・表示されている最終行までのパーセントも表示する。

- @

  現在のファイルについて、コード認識法を順番に変更する。拡張した less には、(環境変数 JLESSCHARSET で選ばれる)デフォルト, japanese、ujis、sjis、iso8、noconv、none という 7 種類のコード認識法がある。

- -

  コマンドラインオプション文字(下記のオプションを参照)のうちの 1 つを続けて入力することで、オプション設定を変更し、新しい設定を解説するメッセージを表示する。ダッシュのすぐ後に^P(CONTROL-P)を入力すると、オプションの設定は変更されるが、メッセージを何も表示しない。オプション文字が、(-b や-h などの)数値変数や (-P や-t などの)文字列変数の場合、オプション文字の後に新しい値が入力できる。値を入力しなかった場合、現在の設定を示すメッセージが表示されるだけで設定は何も変わらない。

- --

  -コマンドと似ているが、1 文字のオプションではなく、長いオプション名(下記のオプションを参照)が使われる。オプション名を打ち込んだ後にリターンを押す必要がある。2 個目のダッシュのすぐ後に^P を入力すると、-コマンドの場合と同じく、新しい設定を説明するメッセージを表示しない。

- -+

  コマンドラインオプション文字のうちの 1 つを続けて入力することで、オプションをデフォルト設定に戻して、新しい設定を表示する。 (″-+X″コマンドは、コマンドラインで″-+X″とするのと同等である。)このコマンドは文字列の値を持つオプションには使えない。

- --+

  -+コマンドと似ているが、1 文字のオプションではなく、長いオプション名が使われる。

- -!

  コマンドラインオプション文字のうちの 1 つを続けて入力することで、オプションをデフォルト設定の「反対」にして、新しい設定を表示する。このコマンドは数字あるいは文字列の値を持つオプションには使えない。

- --!

  -!コマンドと似ているが、1 文字のオプションではなく、長いオプション名が使われる。

- _ (アンダースコア)

  コマンドラインオプション文字のうちの 1 つを続けて入力することで、そのオプションの現在設定を表示する。オプションの設定は変化しない。

- _ (2 つのアンダースコア)

_(アンダースコア)コマンドと似ているが、1 文字のオプションではなく、長いオプション名が使われる。オプション名を打ち込んだ後にリターンを押す必要がある。

- +cmd

  新しいファイルを読み込むたびに、指定したコマンド cmd を毎回実行する。例えば、+G を実行すると less で各ファイルを開いた時に先頭からではなく終端から表示される。

- V

  現在起動している less のバージョンを表示する。

- q、Q、:q、:Q、ZZ

  less を終了する。

以下の 4 つのコマンドが使用できるかは、インストールした方法に依存する。

- v

  現在閲覧しているファイルを編集するためエディタを起動する。エディタとして、環境変数 VISUAL で定義されている値が用いられる。VISUAL が定義されていない場合、環境変数 EDITOR の値が使われる。VISUAL も EDITOR も定義されていない場合、"vi"がデフォルトになる。プロンプトセクションの LESSEDIT に関する話題も参照すること

- ! shell-command

  指定されたシェルコマンドを実行するため、シェルを起動する。コマンド中のパーセント記号(%)は、現在のファイル名で置き換えられる。コマンド中のシャープ記号(#)は、前に読み込んだファイル名で置き換えられる。"!!"は、直前のシェルコマンドを繰り返す。シェルコマンドを伴わない"!"は、単にシェルを起動する。Unix では、シェルは環境変数 SHELL で設定されたものが使われる。設定されていない場合、デフォルトは"sh"である。MS-DOS と OS/2 では、シェルは通常のコマンドプロセッサである。

- | <m> shell-command

  <m>は任意のマーク文字である。入力ファイルのセクションを与えられたシェルコマンドに渡す。渡されるファイルのセクションは、現在の画面の一番上の行から文字でマークされた所までである。ファイルの先頭行と最終行を示すために、<m>をそれぞれ^と$にしてもよい。<m>が.または改行の場合、現在の画面が渡される。

- s filename

  入力をファイルに保存する。このコマンドは入力が一般のファイルでなく、パイプの時のみ有効である。

オプション

コマンドラインオプションを以下に説明する。大部分のオプションは less の実行中に"-"コマンドを用いて変更することが可能である。

大部分のオプションは、「ダッシュと 1 文字のオプション」または「2 つのダッシュと長いオプション名」のどちらかの形式で指定される。長いオプション名は、他のものと区別がつく限り省略できる。例えば、--quit-at-eof は--quit と省略できるが、--qui のように省略することはできない。なぜなら、--quit-at-eof と--quiet が--qui で始まっているからである。大文字のオプション名もある。例えば、--QUIT-AT-EOF のようなものがあり、--quit-at-eof とは区別される。このようなオプション名は、始めの文字を大文字にするだけでよく、それ以降の文字は大文字でも小文字でも構わない。例えば、--Quit-at-eof は --QUIT-AT-EOF と等しい。

環境変数"LESS"と"JLESS"の値もオプションとして使われる。例えば、less を起動するたびに"less-options..."とタイプするのを避けるために、csh では、

setenv LESS "-options"

もし、sh を使っているならば、

LESS="-options"; export LESS

とすればよい。MS-DOS では、クォーテーションは必要ないが、オプション文字列中のパーセント記号を 2 つのパーセント記号に置き換える必要がある。

環境変数はコマンドラインより先に解析されているので、コマンドラインオプションは環境変数 LESS および JLESS を上書きする。もし、あるオプションが環境変数 LESS または JLESS にあっても、コマンドラインオプション"-+"で起動すれば、そのオプションをデフォルトの値にリセットすることができる。

-P や-D オプションのように文字列を後に続けるオプションでは、文字列の終りの印としてダラー記号($)を使わなくてはならない。例えば、MS-DOS で 2 つの-D オプションを設定する場合は、次のように間にダラー記号を入れなくてはならない。

LESS="-Dn9.1$-Ds4.1"

- -?、—help

  このオプションは、less が受け付けるコマンドの概要を表示する(h コマンドと同じ)。(使用しているシェルが?をどのように解釈するかにより、"-¥?"というように?を"で囲む必要があるかもしれない。)

- -a、--search-skip-screen

  画面に表示されている最終行の次の行から検索を開始する。つまり、現在画面に表示されている行中の検索は行わない。デフォルトでは、検索は画面中の第 2 行目(もしくは最後に検索対象が見つかった行のあと。-j オプションを参照)から行われる。

- -bn、--buffers=n

  less が各ファイルに対して使うバッファの数を指定する。各バッファは 1KB で、各ファイルに対してデフォルトでは 10 個のバッファを使う(ただし、ファイルがパイプの場合は例外である。-B オプションを参照)。数字 n で使用するバッファの数を指定する。

- -B、--auto-buffers

  データがパイプから読み込まれる場合、デフォルトではバッファは必要に応じて動的確保される。そのため、大容量のデータがパイプから読み込まれる場合、多くのメモリが確保されてしまう。-B オプションが指定されると、-b オプションで指定されたバッファの数を使うため、パイプに対するバッファの動的確保が行われない。警告:-B オプションを使った場合、ファイルの最も最近閲覧している部分のみしかメモリに保持されず、以前のデータが無くなっているため、表示にエラーが起こる場合もある。

- -c、--clear-screen

  全画面の再描画を、先頭行から下に向かって行わせるようにする。デフォルトでは、全画面の再描画は、画面の最終行からのスクロールによって行われる。

- -C、--CLEAR-SCREEN

  -C オプションは、-c オプションと似ているが、再描画を行う前に画面をクリアする。

- -d、—dumb

  -d オプションは、端末がダムである場合に通常表示されるエラーメッセージの表示させない。ダムとは、画面のクリア、後方へのスクロールといった重要な機能がないことをいう。もしそれらの機能がある場合は、-d オプションでもダム端末上での less の動作は変更されない。

- -Dxcolor、--color=xcolor

  [MS-DOS のみ]表示されるテキストの色を設定する。x は設定するテキストのタイプを表す文字である。n は標準、s は標準出力、d は太字、u は下線、k は点滅である。color は、ピリオドで区切られた数値の組である。1 つ目の数値で文字の前景色、2 つ目の数値で文字の背景色を選ぶ。数値 N は、N.0 と同じである。

- -e、--quit-at-eof

  ファイルの終りに 2 度目に達した場合、自動的に less を終了させる。デフォルトでは、less を終了させる唯一の方法は、″q″コマンドである。

- -E、--QUIT-AT-EOF

  ファイルの終りに 1 度目に達した場合、自動的に less を終了させる。

- -f、--force

  通常のファイルでないものを強制的に開かせる(通常のファイルでないものとは、ディレクトリまたはデバイススペシャルファイルのことである)。また、バイナリファイルを開く場合の警告メッセージも表示しない。デフォルトでは、less は、通常のファイルでないものを開かない。

- -F、--quit-if-one-screen

  最初の画面でファイル全体が表示できる場合、less を自動的に終了させる。

- -g、--hilite-search

  通常、less は、前回の検索とマッチする画面中全ての文字列をハイライト表示する。-g オプションは、前回の検索にマッチした文字列のみをハイライト表示するように変更する。このオプションは、less の動作をデフォルトより多少速くする。

- -G、--HILITE-SEARCH

  -G オプションは、検索コマンドで見つかった文字列をハイライト表示させない。

- -hn、---max-back-scroll=n

  後方に戻る最大行数を指定する。もし、n 行を上回って後方に戻る必要がある場合は、代わりに画面が前方に再描画される(端末が後方に戻る機能を持たない場合は、-h0 を意味する)。

- -I、--ignore-case

  大文字小文字の区別をせず、大文字と小文字は同一とみなして検索をする。このオプションは、検索パターンに大文字が含まれていた場合には無視される。つまり、検索パターンに大文字が含まれていた場合、大文字小文字の区別をした検索をする。

- -I、--IGNORE-CASE

  -i コマンドと似ているが、検索パターンが大文字を含んでいた場合でも、検索は大文字小文字の違いを無視して検索をする。

- -jn、--jump-target=n

  「ターゲット」行とする画面上の行番号を指定する。ターゲット行とは、テキスト検索、タグ検索、行番号へのジャンプ、ファイルのパーセンテージでのジャンプ、マーク位置へのジャンプ、の対象となる行である。画面の行は数字 n で指定する。画面の一番上の行を 1、その次の行を 2、...、と表す。画面の最終行から何行目かを指定する場合は、数値を負に指定する。画面の一番下の行は-1、下から 2 行目は-2、...、と指定する。-j オプションが用いられている時、検索はターゲット行の直後の行から始まる。例えば、"-j4"のとき、ターゲット行は画面の第 4 行目なので、検索は画面の第 5 行目から始まる。

- -J、--status-column

  ステータス欄を画面の左端に表示する。ステータス欄は、-w または-W オプションが有効なときのみ使われる。

- -kfilename、--lesskey-file=filename

  lesskey(1)ファイルとして、指定したファイルを less に開かせて処理させる。複数の-k オプションを指定してもよい。環境変数 LESSKEY または LESSKEY_SYSTEM が設定された場合、もしくは、lesskey ファイルが標準位置(「キー割り当て」セクションを参照)に見つかった場合、それも lesskey ファイルとして使われる。

- -Kcharset

  less に、環境変数 JLESSCHARSET もしくは LESSCHARSET で定義されている文字セットの代わりとして、指定した文字セットを使わせる。

- -m、--long-prompt

  less に、(more のように)詳細なパーセント表示のプロンプトを出させる。デフォルトでは、less は、コロンをプロンプトとして表示する。

- -M、--LONG-PROMPT

  more より、さらに詳細なプロンプトを less に出させる。

- -n、--line-numbers

  行番号を表示させない。(行番号を表示する)デフォルトの設定では、特に、入力ファイルが非常に大きな場合に less の動作が遅くなることがある。-n オプションで行番号非表示にすることで、この問題を避けられる。行番号の使用とは「プロンプトと=コマンドで行番号が表示され、v コマンドで現在の行番号がエディタに渡される」ということである(「プロンプト」セクションにおける LESSEDIT に関する話題を参照すること)。

- -N、--LINE-NUMBERS

  画面の各行の先頭に行番号を表示する。

- -ofilename、--log-file=filename

  less の入力ファイルを指定した名前のファイルにコピーし閲覧する。このオプションは、入力ファイルが一般のファイルではなく、パイプである場合にのみ適用される。ファイルが既に存在する時は、上書きする前に less が確認を求める。

- -Ofilename、--LOG-FILE=filename

  -O オプションは-o に似ているが、既にあるファイルを確認することなく上書きする。
  ログファイルが指定されてない場合、-o と-O オプションは、less のなかで、ログファイルを指定するために使うことができる。ファイル名を指定しない時は、単にログファイル名を表示するだけである。"s"コマンドは、less で、-o を指定するのと同じである。

- -ppattern、--pattern=pattern

  コマンドラインでの-p オプションは、+/pattern を指定するのと同じである。つまり、ファイル中で pattern が最初に現れるところを less の第 1 行目として表示する。

- -Pprompt、--prompt=prompt

  3 つのプロンプトのスタイルを好みによって調整する方法を提供する。このオプションは通常、less コマンドを呼び出すたびに打ち込んだりせずに、環境変数 LESS および JLESS で指定する。そのようなオプションは、環境変数 LESS および JLESS の中で最後のオプションになっているか、もしくは、ダラー記号で終了していなければならない。-Ps の後に文字列を続けるオプションは、デフォルトの(短い)プロンプトをその文字列に変更する。-Pm は中間の(-m)プロンプトを変更する。-PM は長い(-l)プロンプトを変更する。-Ph はヘルプ画面のプロンプトを変更する。-P=は=コマンドで表示されるメッセージを変更する。全てのプロンプトの文字列は、文字列と特別なエスケープシーケンスから構成される。詳細は「プロンプト」セクションを参照すること。

- -q、--quiet、--silent

  比較的「静かな」操作にする。スクロールでファイルの終りを過ぎようとした場合や、ファイルの始まりより前に行こうとした場合でも、端末でベルを鳴らさない。端末に「ビジュアルベル」がある場合は、代わりにそれを使う。不正な文字を打った場合のような、その他のエラーに関してはベルを鳴らす。デフォルトでは、全てのエラーに関して端末のベルが鳴る。

- -Q、--QUIET、--SILENT

完全に「静かな」操作にする。端末のベルは全く鳴らない。

- -r、--raw-control-chars

  「そのままの」制御文字を表示させるようにする。デフォルトでは、制御文字をキャレット表記を使って表示する。例えば、control-A(8 進数 001)は"^A"と表示される。警告:-r オプションが指定されると、less は(制御文字のタイプにどのように画面が反応するかに依存しているために)画面の実際の状況の経過を追うことができない。よって多くの場合、長い行が誤った位置で分割されてしまうといった問題が生じる。

- -R、--RAW-CONTROL-CHARS

  -r と似ているが、可能な場合には画面表示を正しく維持しようとする。このオプションが有効なのは、入力が通常のテキストの場合である。入力には ANSI の「カラー」エスケープシーケンスが含まれていてもよい。このシーケンスは

  ESC [... m

  のような形式で、"..."は"m"以外の 0 個以上の文字である。画面の状況を保つため、全ての制御文字とANSIカラーシーケンスはカーソルを移動させないと仮定している。lessに"m"以外の文字を ANSI カラーエスケープシーケンスの終了文字として認識させることもできる。そのためには、認識させたい終了文字のリストを環境変数 LESSANSIENDCHARS に設定すればよい。

- -s、--squeeze-blank-lines

  連続した空白行を、1 行の空白行にまとめる。nroff の出力を閲覧するときに役立つ。

- -S、--chop-long-lines

  画面幅より長い行を折り返さずに切ってしまう。つまり、長い行の残りの部分を単純に捨ててしまう。デフォルトでは長い行を折り返すので、残りが次の行に表示される。

- -ttag、--tag=tag

  -t オプションの後にはタグ名が続き、そのタグを含むファイルを編集する。このオプションを使うためには、ctags(1)コマンドであらかじめ作られた"tags"と呼ばれるファイルが現在のディレクトリになければならない。このオプションは、新しいファイルを読み込む場合に less のなかで(-コマンドを用いて)指定することもできる。コマンド":t"は、less 中で、-t を指定するのと同じである。

- -Ttagsfile、--tag-file=tagsfile

  "tags"の代わりに使用するタグファイル名を指定する。

- -u、--underline-special

  バックスペースとキャリッジリターンを印刷可能文字として扱う。つまり、これらが入力に現れた場合は端末に送られる。

- -U、--UNDERLINE-SPECIAL

  バックスペース、タブ、キャリッジリターンを制御文字として扱う。つまり、これらの文字は-r オプションで指定されたものとして扱う。

デフォルトで-u と-U のどちらも指定されていない場合、下線文字の隣にあるバックスペースは特別な扱われ方をする。下線された文字は、端末のハードウェア下線機能を使って表示される。さらに、同一の 2 文字の間にあるバックスペースも特別な扱われ方をする。重ね打された文字が端末のハードウェア太字機能を使って表示される。その他の文字に続くバックスペースは削除される。その他のキャリッジリターンは-r オプションで指定されたように扱われる。-u と-U のどちらも有効でない場合に、重ね打ち、もしくは、下線された文字が検索される。

- -V、—version

  less のバージョンナンバーを表示する。

- -w、--hilite-unread

  前方に 1 ページ進んだ場合、最初の「新しい」行を一時的にハイライト表示する。最初の「新しい」行は、前の画面の最下行の次の行である。g または p コマンドの対象となった行もハイライトする。ハイライトは、移動させる次のコマンドがあったときに消される。ステータス行だけをハイライトさせる-J オプションが有効でない限り、行全体がハイライトされる。

- -W、--HILITE-UNREAD

  -w と似ているが、前方に 2 行以上移動した場合に最初の新しい行を一時的にハイライトする。

- -XXX、--mark-wrong-char

  表示できない誤った文字を表示するために、マーク文字(=)が使われるようにする。デフォルトでは、そのような表示できない誤った文字は、バイナリとして表示される。

- -xn、--tabs=n

  タブストップを n 文字に設定する。n のデフォルトは 8 である。

- -X、--no-init

  端末に、termcap 初期化文字列と非初期化文字列を送れないようにする。これは、画面をクリアするときのように非初期化文字列が不必要な場合には、望ましいことがある。

- -yn、--max-forw-scroll=n

  前方に進む最大行数を指定する。もし、n 行を上回って前方に進む必要がある場合は、代わりに画面が再描画される。もし必要であれば、-c と-C オプションは画面の先頭から再描画するために使われる。デフォルトでは、前方移動はスクロールになる。

- -[z]n、--window=n

  スクロールするウインドウのデフォルトの大きさを n 行に変更する。デフォルトは 1 画面分の行数である。z と w コマンドはウインドウの大きさを変更するために使われる。z は、more との移植性のために省略してもよい。n が負の数の時は、現在の画面サイズより n 行小さくウインドウサイズを設定することを意味している。例えば、画面サイズが 24 行の場合、-z-4 はスクロールするウインドウを 20 行に設定することを意味している。さらに、画面サイズが 40 行に変更された場合には、自動的にスクロールウインドウは 36 行に変更される。

- -Z

環境変数 JLESSCHARSET で"japanese"が選択されている場合、UJIS よりも SJIS の優先度を高くする。デフォルトでは、SJIS よりも UJIS の優先度が高い。

- -"cc、--quotes=cc

  ファイル名を引用する文字を変更する。このオプションは、スペースとダブルクォーテーションマークの両方を含む名前をファイルに付けようとする場合に必要となる。-"の後に 1 文字を置いた場合、引用文字が指定した 1 文字に変更される。このとき、スペースを含むファイル名はダブルクォーテーションではなく、この 1 文字で囲まれる。また、-"の後に 2 文字を置いた場合、1 文字目が開クォーテーションで、2 文字目が閉クォーテーションになる。このとき、スペースを含むファイル名の前には開クォーテーション文字を付け、ファイル名の後には閉クォーテーション文字を付ける。引用文字を変更した後でも、このオプションは-"(ダッシュの後にダブルクォーテーション)であることに注意すること。

- -~、--tilde

  通常、ファイルの終端より後の行は 1 個のチルダ(~)を使って表示される。このオプションを使うと、ファイルの終り以降の行は空行として表示される。

- -#、--shift

  RIGHTARROW と LEFTARROW コマンドで水平方向にスクロールするときのデフォルトの移動桁数を指定する。この値を 0 にすると、デフォルトの値は画面幅の半分になる。

- --

  コマンドライン引き数"--"は、オプション引き数の終りの印である。この後の、いかなる引き数もファイル名として解釈される。このオプションは、名前が"-"または"+"で始まるファイルを閲覧する場合に役立つ。

- +

  あるコマンドラインオプションが+で始まる場合、その残りは、less の初期化コマンドとして渡される。例えば、+G では、ファイルの先頭ではなく終端を表示して less を起動させる。そして、オプション+/xyz では、ファイル中で"xyz"が始めて現れる場所から起動させる。特殊な場合として、+<number>は+<number>g と同じ働きをする。つまり、このオプションでは指定された行数から表示が始まる (しかし、上の"g"コマンドの注意書きを参照すること)。オプションが++で始まっている場合、初期化コマンドは閲覧している一番始めのファイルだけでなく、全てのファイルに対して適用される。以前説明した"+"コマンドも、全てのファイルに対する初期化コマンドの設定(および変更)に使われる。

ラインエディット

画面の一番下で(例えば、:e コマンドのファイル名や、検索コマンドのパターンといった)コマンドライン入力の場合、いくつかキーがコマンドラインを編集するのに使われる。大部分のコマンドには、[大括弧]中の別形式がある。これは、ある種のキーボードでキーが存在しない場合に使用できる(大括弧の中の形式は、MS-DOS 版では機能しない)。それらの特殊キーは、^V や^A といった「リテラル化」文字を先に入

力することで、そのままの文字として入力することができる。2 つのバックスラッシュを入力することで、バックスラッシュ自身も文字として入力することができる。

- LEFTARROW [ESC-h]
  カーソルを 1 文字分左へ移動する。
- RIGHTARROW [ESC-l]
  カーソルを 1 文字分右へ移動する。
- LEFTARROW [ESC-b、ESC-LEFTARROW]
  (CONTROL と LEFTARROW を同時に入力すると)カーソルを 1 単語分左へ移動する。
- RIGHTARROW [ESC-w、ESC-RIGHTARROW]
  (CONTROL と RIGHTARROW を同時に入力すると)カーソルを 1 単語分右へ移動する。
- HOME [ESC-0]
  カーソルを行頭へ移動する。
- END [ESC-$]
  カーソルを行末へ移動する。
- BACKSPACE
  カーソルの左にある文字を消去する。コマンドラインが空の場合は、コマンドをキャンセルする。
- DELETE、[ESC-x]
  カーソルの下にある文字を消去する。
- BACKSPACE [ESC-BACKSPACE]
  (CONTROL と BACKSPACE を同時に入力すると)カーソルの左にある単語を消去する。
- DELETE [ESC-X、ESC-DELETE]
  (CONTROL と DELETE を同時に入力すると)カーソルの下にある単語を消去する。
- UPARROW [ESC-k]
  (ヒストリにある)前のコマンドラインを呼び出す。
- DOWNARROW [ESC-j]
  (ヒストリにある)次のコマンドラインを呼び出す。
- TAB
  カーソルの左にある部分的なファイル名を補完する。複数のマッチするファイル名がある時は、最初にマッチしたファイル名がコマンドラインに出される。TAB を繰り返し打つと、マッチしたファイル名が順番に表示される。補完したファイル名がディレクトリの場合、″/″がファイル名に付加される(MS-DOS では、″¥″が付加される)。ディレクトリ名に付加する文字の指定するために、環境変数 LESSSEPARATOR を使うことができる。
- BACKTAB [ESC-TAB]
  TAB と同様であるが、マッチしたファイル名を逆順に表示する。
- L

カーソルの左にある部分的なファイル名を補完する。複数のマッチするファイル名がある時は、全てのマッチしたファイル名が入力される。

- U(Unix)、ESC(MS-DOS)

  コマンドライン全体を消去する。コマンドが空の場合は、コマンドをキャンセルする。Unix において、一行消去のための文字を^U 以外に変更している場合、その文字が^U の代わりに使われる。

キー割り当て

lesskey ファイルを生成する lesskey(1)というプログラムを用いて、独自の less コマンドを定義できる。このファイルは、コマンドキーとそれに関係づけられたアクションを指定する。ラインエディットキー(「ラインエディット」セクションを参照)の変更や環境変数の設定のために lesskey を使うこともできる。環境変数 LESSKEY が設定されている場合、less は lesskey ファイル名としてその値を使う。設定されていない場合、less は、標準の位置にある lesskey ファイルを探す。Unix の場合、less は"$HOME/.less"というファイルを探す。MS-DOS と Windows の場合、less は"$HOME/_less"という lesskey ファイルを探す。存在しない場合は、環境変数 PATH で指定されている全てのディレクトリの下にある"_less"という lesskey ファイルを探す。OS/2 の場合、less は"$HOME/less.ini"という lesskey ファイルを探す。存在しない場合は、環境変数 INIT で指定されている全てのディレクトリの下にある"less.ini"ファイルを探す。それでもない場合は、環境変数 PATH で指定されている全てのディレクトリの下にある"less.ini"ファイルを探す。詳細は lesskey のマニュアルページを参照すること。

キー割り当てを提供するために、システム共通の lesskey ファイルを設定することもできる。ローカルな lesskey ファイルとシステム共通の lesskey ファイルの両方でキーが定義された場合、ローカルなファイルにあるキー割り当ての方がシステム共通のキー割り当てより優先される。環境変数 LESSKEY_SYSTEM が設定されると、less はそれをシステム共通の lesskey ファイルの名前として使う。この環境変数が設定されていない場合、less は次に示す場所を標準的なシステム共通の lesskey ファイルの場所として探す。Unix では、システム共通の lesskey ファイルは/usr/local/bin/.sysless である。(しかし、less バイナリ用ディレクトリが/usr/local/bin とは異なる場所にビルドされていた場合、そのディレクトリに.sysless ファイルがある。)MS-DOS と Windows では、システム共通の lesskey ファイルは c:¥_sysless である。OS/2 では、システム共通の lesskey ファイルは c:¥_sysless.ini である。

入力プリプロセッサ

less のための「入力プリプロセッサ」を定義することができる。less がファイルを開く前に、入力プリプロセッサで入力ファイルの内容の表示の仕方を変更することができる。入力プリプロセッサに渡される。入力プリプロセッサは、ファイルの内容を代替ファイルと呼ばれる別ファイルに書き出す単純な実行可能プログラム(もしくは、シェルスクリプト)である。代替ファイルの内容がオリジナルファイルの内容の代わりに表示される。しかし、ユーザーにとってはオリジナルファイルが開かれているかのように見える。less は現在

の代替ファイルの名前としてオリジナルファイルの名前を表示する。

入力プリプロセッサは、ユーザーによって入力されるオリジナルファイル名を 1 つのコマンドライン引き数として受け付ける。そして、代替ファイルを生成し終えると、代替ファイル名を標準出力に表示する。入力プリプロセッサが代替ファイル名を出力しない場合、less は標準としてオリジナルファイルを用いる。入力プリプロセッサは、標準入力を閲覧する場合には呼び出されない。入力プリプロセッサを設定するためには、入力プリプロセッサを呼び出すコマンドラインを環境変数 LESSOPEN に設定する。このコマンドラインには、入力プリプロセッサコマンドが呼び出されるときに、ファイル名に置き換えられる文字列"%s"を含んでいなければならない。

less がそのようにして開いたファイルを閉じる時には、入力ポストプロセッサと呼ばれるもう 1 つのプログラムが呼び出される。このプログラムは、(LESSOPEN で開かれた代替ファイルを消去するといった)全ての必要な後処理をする。このプログラムは、ユーザーによって入力されたオリジナルファイル名と代替ファイル名の 2 つを引き数として受け付ける。入力ポストプロセッサを設定するためには、入力ポストプロセッサを呼び出すコマンドラインを環境変数 LESSCLOSE に設定する。入力ポストプロセッサコマンドはファイル名に置き換えられる文字列"%s"を 2 つ含んでいる。1 つ目はファイルのオリジナルの名前に置き換えられ、2 つ目は LESSOPEN の出力である代替ファイルの名前に置き換えられる。

例えば、次の 2 つのスクリプトにより、多くの Unix システムでは、圧縮されているファイルを展開せずに less でファイルを閲覧することができる。

```
lessopen.sh:
#! /bin/sh
case "$1" in
*.Z) uncompress -c $1  >/tmp/less.$$  2>/dev/null
  if [ -s /tmp/less.$$ ]; then
        echo /tmp/less.$$
  else
        rm -f /tmp/less.$$
  fi
  ;;
esac

lessclose.sh:
#! /bin/sh
rm $2
```

このスクリプトを使うためには、2 つを実行可能な場所に置き、LESSOPEN="lessopen.sh %s"、LESSCLOSE="lessclose.sh %s %s"というように環境変数を設定する。他の圧縮ファイルを受け付けるような、更に複雑な LESSOPEN と LESSCLOSE スクリプトを書くことも可能である。

ファイルのデータを代替ファイルに書き出さず、そのまま、less にパイプするような入力プリプロセッサを設定することも可能である。こうすることにより、閲覧する前に圧縮ファイル全体を展開するのが避けられる。このような働きをする入力プリプロセッサは、入力パイプと呼ばれる。入力パイプは、代替ファイル

名を標準出力に表示する代わりに、代替ファイルの内容全てを標準出力に書き出す。入力パイプが標準出力に何も書き出さない場合、代替ファイルは生成されず、less は普通にオリジナルファイルを使う。入力パイプを使う場合は、入力プリプロセッサが入力パイプであることを知らせるために、環境変数 LESSOPEN の最初の文字を、縦棒(|)に設定する。

例えば、多くの Unix システムで、次のスクリプトは上で説明したサンプルスクリプトと同じ働きをする。

```
lesspipe.sh:
#! /bin/sh
case "$1" in
*.Z) uncompress -c $1  2>/dev/null
  ;;
esac
```

このスクリプトを使うためには、実行可能な場所に置いて、LESSOPEN="| lesspipe.sh%s"と設定する。入力パイプが使われる場合も、ポストプロセッサ LESSCLOSE を使っても良いが、削除する代替ファイルがないので多くの場合必要ない。このスクリプトで、LESSCOLSE のポストプロセッサに渡される代替ファイル名は"-"である。

国際化文字セット

入力ファイルには、3 タイプの文字が含まれている。

- ノーマル文字
  画面に直接表示できる文字。
- 制御文字
  画面には直接表示すべきでないが、(バックスペースやタブのように)普通のテキストファイル中にある文字。
- バイナリ文字
  画面には直接表示すべきでなくテキストファイル中にはない文字。

「文字セット」とは、簡潔にいうと、どの文字がノーマル文字・制御文字・バイナリ文字とされるか、ということである。環境変数 JLESSCHARSET と LESSCHARSET が文字セットを選択するのに使われる。less のプログラム中で、この 2 つの環境変数に違いはないが、環境変数 JLESSCHARSET を使うことを勧める。なぜなら、環境変数 LESSCHARSET に拡張文字セットを設定していると、拡張文字セットの使えない less がエラーを起こすからである。この環境変数に設定できる値を以下に示す。

- ascii
  BS、TAB、NL、CR、formfeed が制御文字である。32 から 126 の値を持つ文字がノーマルで、その他がバイナリである。
- iso8859
  ISO8859 文字セットを選択する。160 から 255 までの文字がノーマルとして扱われる以外は、

ASCII と同じである。

- latin1

  iso8859 と同じ。

- dos

  MS-DOS 用の文字セットを選択する。

- ebcdic

  EBCDIC 文字セットを選択する。

- koi8-r

  ロシア語文字セットを選択する。

- next

  NeXT 計算機用の文字セットを選択する。

- utf-8

  ISO10646 文字セットの UTF-8 エンコーディングを選択する。

JLESSCHARSET にのみ設定可能な値を以下に示す。

- iso7

  7 ビットを仮定した ISO 2022 コード拡張による複数文字セット。128 から 225 の値を持つ文字がバイナリとして扱われる。less の実装レベルは、ISO 2022 の level 3 である。

- iso8

  8 ビットを仮定した ISO 2022 コード拡張による複数文字セット。less の実装レベルは、ISO 2022 の level 3 である。

- jis

  7 ビットを仮定した ISO 2022 コード拡張による日本語文字セットのみ。

- ujis

  文字が 32 から 127 の値を持つ場合、ASCII 文字列を仮定する。文字が 162 から 254 の値を持つ場合、JISX 0208 文字セット・JISX0201 文字セットの右半分・UJIS コーディングをした JISX 0212 文字セットを仮定する。それ以外の文字はバイナリとして扱われる。

- euc

  ujis と同じ。

- sjis

  文字が 32 から 127 の値を持つ場合、ASCII 文字列を仮定する。文字が 128 から 252 の値を持つ場合、JISX0208 文字セットと JISX0201 文字セットの右半分を仮定する。それ以外の文字はバイナリとして扱われる。

- japanese

  入力は全ての日本語文字セット、jis、ujis、sjis を仮定する。less は jis のみを出力する。

日本語には、いくつかのコードセット(文字セットではない)が存在する。よって、less は、それを正しく表示

するためにコードセット間の変換をしなければならない。変換をさせるために設定できる JLESSCHARSET の値を以下に示す。

- ujis-iso7

  入力は ujis と iso7 を仮定する。less は iso7 のみを出力する。

- euc-iso7

  ujis-iso7 と同じ。

- sjis-iso7

  入力は sjis と iso7 を仮定する。less は iso7 のみを出力する。

- ujis-jis

  入力は ujis と jis を仮定する。less は jis のみを出力する。

- euc-jis

  ujis-jis と同じ。

- sjis-jis

  入力は sjis と jis を仮定する。less は jis のみを出力する。

- jis-ujis

  入力は jis と ujis を仮定する。less は ujis のみを出力する。

- jis-euc

  jis-ujis と同じ。

- jis-sjis

  入力は jis と sjis を仮定する。less は sjis のみを出力する。

- japanese-iso7

  入力は japanese と iso7 を仮定する。less は iso7 のみを出力する。

- japanese-jis

  入力は japanese を仮定する。less は jis のみを出力する。japanese と同じ。

- japanese-ujis

  入力は japanese を仮定する。less は ujis のみを出力する。

- japanese-euc

  japanese-ujis と同じ。

- japanese-sjis

  入力は japanese を仮定する。less は sjis のみを出力する。

- ujis-sjis

  入力は ujis を仮定する。less は sjis のみを出力する。

- euc-sjis

  ujis-sjis と同じ。

- sjis-ujis

  入力は sjis を仮定する。less は ujis のみを出力する。

- sjis-euc

  sjis-ujis と同じ。

文字セットを設定するもう一つ方法は、環境変数 LANG を使うことである。"ja_JP"または、"japan"で始まる環境変数 LANG であるとき、less は、全ての日本語コード文字を日本語文字セットとして読み、環境変数 LANG の残りの部分で出力のコーディングを指定する。

ISO 2022コード拡張テクニックでは、多くの文字セットを簡単に表示させるために4つのプレーンを定義している。環境変数 JLESSPLANESET によりデフォルトのプレーンを設定する。環境変数 JLESSPLANESET が、"japanese"、"ujis"、"euc"のとき、less は、g1 プレーンを JISX 0208、g2 プレーンを JISX 0201 の右半分、g3 プレーンを JISX 0212 として扱う。環境変数 JLESSPLANESET が、"latin1"、"latin2"、"latin3"、"latin4"、"greek"、"alabic"、"hebrew"、"cyrillic"、"latin5"のとき、less は、g1 プレーンを ISO 8859 の一つとして扱う。それ以外の場合、less は、環境変数 JLESSPLANESET を設定のための実際のエスケープシーケンスとして解析しようと試みる。解析の際、JLESSPLANESET における"¥e"はエスケープコードとして扱われる。

less は、ISO 2022 コード拡張テクニックに含まれる文字セットのほとんど全てのエスケープシーケンスを理解する。文字セットを選択するためのエスケープシーケンスは多数存在する。一方で、less は、文字列セットを選択するために、6 つのエスケープシーケンスしか出力しない。これは、less が端末と端末エミュレーターにとって使いやすいということである。

キーボード入力のための特別な「文字セット」もある。環境変数 JLESSKEYCHARSET は、この文字セットを設定するために使われる。JLESSKEYCHARSET が持つことのできる値は、環境変数 JLESSCHARSET と同じである。

LESSCHARSET 環境変数が設定されていない場合、デフォルトの文字セットは latin1 である。しかし、環境変数 LC_ALL、LC_CTYPE、LANG のいずれかに"UTF-8" という文字列がある場合、デフォルトの文字セットは utf-8 になる。

特殊なケースとして、LESSCHARSET で指定できる限定された文字セットではなく、他の文字セットを使うようにlessを調整したいという場合もある。このようなとき、環境変数LESSCHARDEF が文字セットを定義するのに使われる。この変数の定義において、文字"."はノーマルキャラクタ、文字"c"は制御文字、文字"b"はバイナリキャラクタを表す。また、10 進数は定義文字の繰り返しを示す。例を挙げると、"bccc4b."は値 0 の文字がバイナリ、1、2、3 はコントロール、4、5、6 はバイナリ、8 はノーマルキャラクタという意味である。最後に文字タイプが指定された文字より後の全ての文字は、最後の文字と同じタイプとみなされるので、9 から 255 までの値を持つ文字はノーマルとなる。(これは 1 つの例であるので、必ずしも実際の文字セットを表しているわけではない。)

下の表は、各 LESSCHARSET と等しい LESSCHARDEF の値を示している。

- ascii

  8bcccbcc18b95.b

- dos

  8bcccbcc12bc5b95.b.

- ebcdic

  5bc6bcc7bcc41b.9b7.9b5.b..8b6.10b6.b9.7b

  9.8b8.17b3.3b9.7b9.8b8.6b10.b.b.b.

- iso8859

  8bcccbcc18b95.33b.

- koi8-r

  8bcccbcc18b95.b128.

- latin1

  8bcccbcc18b95.33b.

- next

  8bcccbcc18b95.bb125.bb

LESSCHARSET と LESSCHARDEF のどちらも設定されていない場合でも、システムが setlocale インターフェースをサポートしていれば、less は文字セットを決定するのに setlocale を使用する。setlocale は環境変数 LANG もしくは LC_CTYPE で制御される。

制御文字とバイナリキャラクタは目立たせて(反転して)表示される。これらの文字は、可能であるならば(control-A を^A というように)キャレット表記で表示される。キャレット表記は、0100 ビットを反転した結果が印刷可能文字になるときのみ使われる。キャレット表記できないときは、角型括弧<>で囲まれた 16 進数で表す。このフォーマットは、環境変数 LESSBINFMT を設定することで変えられる。LESSBINFMT は"*"で始まり、もう1つの文字が表示属性を選択する。"*k"は点滅、"*d"は太字、"*u"は下線、"*s"は反転、"*n"は標準である。LESSBINFMT が"*"で始まっていないときは、標準の属性を仮定する。LESSBINFMT の残りの部分は、1 つの printf スタイルのエスケープシーケンス(%と x、X、o、d など)を含む文字列である。LESSBINFMT が"*u[%x]"の場合、バイナリキャラクタは大括弧で囲まれた 16 進数に下線をして表示される。LESSBINFMT が指定されていないときのデフォルトは、"*s<%X>"に指定されている。

プロンプト

-P オプションにより、プロンプトを自分の好みに調整することができる。-P オプションに与えられた文字列は、指定したプロンプト文字列を置き換える。文字列中のある文字は特殊文字として解釈される。プロンプト機構は柔軟性を持たせるために少々複雑になっているが、一般のユーザーは、個人用プロンプト文字列の作り方を詳細に理解する必要はない。

パーセント記号は、その後に何の文字が続くかによって展開される。

- %bX

  現在の入力ファイルへのバイトオフセットで置き換えられる。b の後には、(X で示される)1 文字が続き、どの行のバイトオフセットを使うかを指定する。"t"のときは、画面の先頭行のバイトオフセットが使われる。"m"は真中の行、"b"は最終行、"B"は最終行のすぐ次の行のバイトオフセットを使うことを意味する。そして、"j"の場合には-j オプションで指定したターゲット行のバイトオフセットを使うことを意味している。

- %B

  現在の入力ファイルの大きさに置き換えられる。

- %c

  画面左端に現れるテキストのカラム番号に置き換えられる。

- %dX

  対象となる行が入力ファイルにおいて何ページ目であるかに置き換えられる。%b オプションと同じように、X で対象とする行を決定する。

- %D

  入力ファイルのページ数に置き換えられる。つまり、入力ファイルの最終行のページ番号に等しい。

- %E

  エディタの名前(環境変数 VISUAL、VISUAL が定義されていないときは環境変数 EDITOR)に置き換えられる。以降の、JLESSEDIT の特徴に関する話題を参照。

- %f

  現在の入力ファイル名に置き換えられる。

- %I

  入力ファイルのリスト中における現在のファイルのインデックスで置き換えられる。

- %lX

  入力ファイルで何行目にいるかで置き換えられる。対象となる行は、%b オプションと同じく X で決定される。

- %L

  入力ファイルの最終行の行番号で置き換えられる。

- %m

  入力ファイルの合計数で置き換えられる。

- %pX

  バイトオフセットで計算して、現在の入力ファイルで何パーセントの場所にいるかで置き換えられる。対象となる行は、%b オプションと同じく X で決定される。

- %PX

  行番号で計算して、現在の入力ファイルで何パーセントの場所にいるかで置き換えられる。対象となる行は、%b オプションと同じく X で決定される。

- %s

  %B と同じ。

- %t

  後に続くスペースを取り除かせる。通常は文字列の後に使われるが、どこに置いてもよい。

- %x

  ファイルリストのうちの次の入力ファイル名で置き換えられる。

- %K

  入力ファイルの中で最も最近現れた、非 ASCII 文字セットまたは非 ASCII コードセットで置き換えられる。

もし、(入力がパイプのためファイルサイズが分からない場合など)項目が不明のときは、代わりにクエスチョンマークが表示される。

プロンプト文字列のフォーマットは、ある条件に応じて変更することができる。クエスチョンマークとその後に続く 1 文字で、"IF"のように働き、どのような文字が続くかで、条件文が評価される。条件文が真ならば、クエスチョンマークと条件文字の後に続く文字列からピリオドまでがプロンプトの中に表示される。条件文が偽なら、文字列はプロンプトに表示されない。クエスチョンマークとピリオドの間にあるコロンは、"ELSE"として働き、コロンとピリオドの間にある文字列は、IF 文が偽のときのみプロンプト文字列に組み入れられる。(クエスチョンマークの後の)条件文字は次のようなものがある。

- ?a

  プロンプトに既に文字列が含まれているときに真。

- ?bX

  指定した行のバイトオフセットが既知の場合に真。

- ?B

  現在の入力ファイルの大きさが既知の場合に真。

- ?c

  水平方向にテキストがシフトしている(%c が 0 でない)場合に真。

- ?dX

  指定された行のページ番号が既知の場合に真。

- ?e

  ファイルの終りのときに真。

- ?f

  入力ファイル名があるとき(入力がパイプでないとき)に真。

- ?lX

  指定した行の行番号が既知の場合に真。

- ?L

  ファイルの最終行の行番号が既知の場合に真

- ?m

  2 つ以上の入力ファイルがある場合に真。

- ?n

  新しい入力ファイルの最初のプロンプトのとき真。

- ?pX

  指定した行の現在の入力ファイルでのバイトオフセットで計算したパーセントが既知のとき真。

- ?PX

  指定した行の現在の入力ファイルでの行番号で計算したパーセントが既知のとき真。

- ?s

  "?B"と同じ。

- ?x

  次の入力ファイルがあるとき(現在の入力ファイルが最後のファイルでないとき)真。

特殊文字(クエスチョンマーク、コロン、ピリオド、パーセント、バックスラッシュ)以外の全ての文字がプロンプトの実際に表示される部分になる。特殊文字をプロンプトに表示させるには、その文字の前にバックスラッシュを置けばよい。

プロンプトの例をいくつか挙げる。

```
?f%f:Standard input.
```

このプロンプトは、ファイル名が既知であれば表示する。わからないときは"Standard input"と表示する。

```
?f%f .?ltLine %lt:?pt%pt¥%:?btByte %bt:-...
```

このプロンプトは、ファイル名が既知であれば、ファイル名を表示する。行番号が既知のときには、ファイル名に続けて行番号を表示する。もし行番号が既知でなく、パーセントが既知のときは、パーセントを表示する。パーセントも既知でなく、バイトオフセットが既知のときは、バイトオフセットを表示する。バイトオフセットも既知でないときは、ダッシュを表示する。各クエスチョンマークにピリオドがどのように対応しているかに注意すること。また、%pt の後の%を実際に表示させるために、バックスラッシュでエスケープしていることにも注意すること。

```
?n?f%f .?m(file %i of %m) ..?e(END) ?x- Next¥: %x..%t
```

このプロンプトがファイルにおける最初のプロンプトのときファイル名を表示する。さらに、2 つ以上のファイルがあるときは、"file N of N"というメッセージを加える。そして、ファイルの終りに達したときは、文字列"(END)"を表示し、引き続いて次のファイルがあるときはそのファイル名を表示する。最後に、後に続くスペースを切り詰める。これはデフォルトで使われているプロンプトである。参考として、(-mと-Mオプションに対応する)2 つのデフォルトプロンプトを挙げる。ここで 2 行に分けたのは、読みやすさの為だけで実際は分けない。

```
?n?f%f .?m(file %i of %m) ..?e(END) ?x- Next¥: %x.:
```

```
?pB%pB¥%:byte %bB?s/%s...%t

?f%f .?n?m(file %i of %m) ..?ltlines %lt-%lb?L/%L. :
byte %bB?s/%s. .?e(END) ?x- Next¥: %x.:?pB%pB¥%..%t
```

そして、次に挙げるのは、=コマンドで出されるデフォルトのメッセージである。

```
?f%f .?m(file %i of %m) .?ltlines %lt-%lb?L/%L. .
byte %bB?s/%s. ?e(END) :?pB%pB¥%..%t
```

プロンプト展開の機能は、他の目的でも使われる。環境変数 LESSEDIT が定義されている場合、この変数は v コマンドで実行されるコマンドとして使われる。LESSEDIT の文字列は、プロンプト文字列と同じ方法で展開される。LESSEDIT のデフォルトの値を以下に示す。

```
%E ?lm+%lm. %f
```

この文字列は、「エディタ名」・「+と行数」・「ファイル名」に展開される。指定したエディタが、"+行数"という書式を受け付けない場合や、呼び出しの書式が違う場合は、このデフォルトの LESSEDIT を修正することができる。

セキュリティ

環境変数 LESSSECURE が 1 に設定されている場合、less は「安全な」モードで実行される。この場合、次のことができない。

- !
  シェルコマンド
- |
  パイプコマンド
- :e
  ファイルの読み込みコマンド
- v
  編集コマンド
- s
  -o ログファイル
- -k
  lesskey ファイルを使う
- -t
  タグファイルを使う
- ファイル名の中の、*といった、メタキャラクタ
- ファイル名補完(TAB、^L)

less をいつも「安全な」モードでしか実行できないようにコンパイルすることも可能である。

環境変数

環境変数は、通常と同じくシステム環境で設定でき、lesskey(1)ファイルでも指定できる。環境変数が複数で指定されている場合、ローカルな lesskey ファイルで定義されている変数は、システム環境で定義されている変数より優先される。また、システム環境で定義されている変数は、システム共通の lesskey ファイルで定義されている変数より優先される。

- COLUMNS
  画面の 1 行あたりの文字数を設定する。環境変数 TERM で設定された値より優先される (TIOCGWINSZ や WIOCGETD をサポートするウインドウシステムの場合、ウインドウシステムが持つ画面サイズの方が、環境変数 LINES と COLUMNS より優先される)。
- EDITOR
  (v コマンドで使われる)エディタの名前。
- HOME
  (Unix で lesskey ファイルを探すのに使われる)ユーザーのホームディレクトリ名。
- HOMEDRIVE、HOMEPATH
  環境変数 HOME が設定されていない場合、環境変数 HOMEDRIVE と HOMEPATH を足したものがユーザーのホームディレクトリ名となる (Windows バージョンのみで有効)。
- INIT
  (OS/2 で lesskey ファイルを探すのに使われる)ユーザーの init ディレクトリ名。
- LANG
  文字セットを決定するための言語。
- LC_CTYPE
  文字セットを決定するための言語。
- LESS
  less に自動的に渡されるオプション。
- JLESS
  環境変数 LESS と同じ。
- LESSANSIENDCHARS
  ANSI カラーシーケンスの終りとして使われる文字(デフォルトは"m")。
- LESSBINFMT
  印字可能文字でもなく、制御文字でもない文字を表示する際のフォーマット。
- LESSCHARDEF
  文字セットを定義する。
- JLESSCHARSET
  あらかじめ定義された文字セットを選択する。
- LESSCHARSET

JLESSCHARSET が定義されていない場合に、あらかじめ定義された文字セットを選択する。

- JLESSKEYCHARSET

  キーボート入力のためのあらかじめ定義された文字セットを選択する。

- JLESSPLANESET

  あらかじめ定義された ISO2022 コード拡張テクニックのプレーンセットを選択する。

- LESSCLOSE

  (オプションの)入力ポストプロセッサを呼び出すためのコマンドライン。

- LESSECHO

  lessecho プログラムの名前(デフォルトは、"lessecho")。lessecho プログラムは、Unix システム上のファイル名を、*や?といったメタキャラクタで展開するのに必要である。

- LESSEDIT

  (vコマンドで使われる)エディタのプロトタイプ文字列。「プロンプト」セクションでの話題を参照。

- LESSKEY

  デフォルトの lesskey(1)ファイルの名前。

- LESSKEY_SYSTEM

  デフォルトで使われるシステム共通の lesskey(1)ファイルの名前。

- LESSMETACHARS

  シェルに「メタキャラクタ」として解釈される文字のリスト。

- LESSMETAESCAPE

  less がシェルにコマンドを送る際にメタキャラクタの前に付加するプレフィックス。LESSMETAESCAPE が空の文字列であるとき、メタキャラクタを含むコマンドはシェルに送られない。

- LESSOPEN

  (オプションの)入力プリプロセッサを起動するためのコマンドライン。

- LESSSECURE

  less を「安全な」モードで実行させる。「セキュリティ」セクションでの話題を参照すること。

- LESSSEPARATOR

  ファイル名補完においてディレクトリ名に付加される文字列。

- LINES

  画面の行数を設定する。環境変数 TERM で指定された行数より優先される。(TIOCGWINSZ や WIOCGETD をサポートするウインドウシステムの場合、ウインドウシステムが持つ画面サイズの方が、環境変数 LINES と COLUMNS より優先される)。

- PATH

  (MS-DOS と OS/2 で lesskey ファイルを探すのに使われる)ユーザーの検索パス。

- SHELL

  !コマンドを実行したり、ファイル名を補完するのに使われるシェル。

- TERM

  less が実行される端末のタイプ。

- VISUAL

  (v コマンドで使われる)エディタの名前。

関連項目

lesskey(1)

警告

(-P オプションで変更されない限り)=コマンドとプロンプトは、画面の先頭行と最終行の行番号を表示する。しかし、バイト数とパーセントは画面最終行の次行について表示する。

:e コマンドが 2 つ以上のファイルに対して使われ、ファイルのうちの 1 つが既に閲覧されている場合、新しいファイルが予期しない順番でリストに入れられる。

ある種の古い端末(いわゆる「マジッククッキー」端末)では、検索のときのハイライトが表示エラーを起こす。そのような端末では、この問題を避けるため、検索の際のハイライトがデフォルトで不可に設定されている。

検索の際のハイライト表示が設定されている場合に、検索パターンが^で始まっていると、マッチした文字列以外の部分までハイライトされることがある (POSIX 正規表現パッケージを使って less がコンパイルされていれば、この問題は起こらない)。

setlocale が 0 から 31 の値を持つ ASCII 文字をバイナリキャラクタでなく制御文字とするシステムもある。そのため、less がある種のバイナリファイルをバイナリでない通常のファイルとして扱かってしまう。この問題を解決するには、環境変数 LESSCHARSET を"ascii"(もしくは、何か適切な文字セット)に設定すればよい。

6.2.5.16 指定した文字列で始まる行を表示する (look)

書式

```
look [-dfa] [-t termchar] string [file]
```

説明

look は、file の各行のうち、string で指定された文字列で始まっている行を表示する。look は二分検索を使っているため、file で指定するファイルはソートしておかなければならない。(look で-d オプションを使うときにはソートするときの sort(1)にも-d を使い、-f オプションを使うときは sort(1)でも-f を使うこと)

file が指定されなかった場合は/usr/share/dict/word が使用され、アルファベットと数字だけで比較され、アルファベットの大文字小文字の違いは無視される。

オプション

- -d
  辞書で使用されているアルファベットと数字だけを比較対象とする。
- -f
  アルファベットの大文字、小文字を区別しない。
- -a
  別の辞書/usr/share/dict/web2 を使用する。
- -t
  文字列の終端文字を指定する。つまり、string のうち termchar が最初に出てくるところまで(termchar を含む)が比較の対象となる。

返り値

look は、指定の文字列で始まる行が見つかった場合は 0 を、見つからなかった場合は 1 を、エラーが起きた場合は 2 以上の値を返す。

ファイル

- /usr/share/dict/words
  デフォルトで使用される単語ファイル
- /usr/share/dict/web2
  もう一つの単語ファイル

関連項目

grep(1)、sort(1)

6.2.5.17 多言語対応のファイルビューワ (lv)

書式

```
lv、lgrep
lv -h
lv -V
lv [-[+]acdfgiklmnqsuvz]
[-Acoding-system] [-Icoding-system] [-Kcoding-system]
[-Ocoding-system] [-Pcoding-system] [-Dcoding-system]
```

```
[-Ssseq] [-Srseq] [-Sbseq] [-Suseq] [-Shseq]
[-Tnumber] [-Wwidth] [-Hheight] [-E'editor'] [-+]
[-] (grep pattern) [files ...]
```

説明

多言語対応ファイルビューワ

lv は多言語対応の強力なファイルビューワである。lv は less などの有名な UNIX のファイルビューアに似ているため、新しく操作方法を学ぶ必要はない。lv は MSDOS ANSI ターミナルとほとんどの UNIX プラットフォームで使用することができる。lv は開発途上のソフトウェアである。lv をより良いものにするためにも、私たちはあなたのフィードバックを歓迎する。

複数のコーディングシステム

lv は多くのコーディングシステムによって、多くの言語のデコード/エンコードを行うことができる。例えば、iso-2022-jp のような ISO 2022 を元にしたエンコーディングシステムや、euc-japan のような EUC (Extended Unix Code)を扱うことが可能である。さらに、shift-jis、big5、HZ のようなローカライズされたコーディングシステムにも対応する。lv はファイルビューアとしてだけでなく、nkf(1)や tcs(1)のようなコーディングシステムの変換フィルタとしても利用することができる。

多言語対応の正規表現と grep

lv はマルチバイトの正規表現を扱うことができる。また、多言語対応の grep 機能である lgrep も提供する。パターンマッチはキャラクターセットレベルで実施されるため、もちろん、例えば ISO 2022 のストリームから EUC の断片を発見することができる。

Unicode 対応

lv は UTF-7 や UTF-8 などのユニコードを扱うことができる。また、ユニコードと他のキャラクタセット完のコードポイント変換も行うことができる。ユニコードを通して変換することで、ユニコードや外国のテキストを好みのキャラクタセットでターミナルに表示させることができる。(しかし、MSDOS バージョンはユニコード機能を有さない。)

ANSI エスケープシーケンス対応

lv はテキスト装飾のための ANSI エスケープシーケンスを認識する。他のソフトウェアによって作成され、色づけされたソースコードなどのような ANSI 装飾文字列を ANSI ターミナル上で表示させることができる。

完全なオリジナル

lv は完全なオリジナルソフトウェアであり、less や grep、その他のプログラムのコードは含んでいない。

オプション

- -A<coding-system>

  全てのコーディングシステムを coding-system に設定する。

- -I<coding-system>

  入力コーディングシステムを coding-system に設定する。

- -K<coding-system>

  キーボードコーディングシステムを coding-system に設定する。設定されていなければ、出力コーディングシステムが設定される。

- -O<coding-system>

  出力コーディングシステムを coding-system に設定する。

- -P<coding-system>

  pathname コーディングシステムを coding-system に設定する。

- -D<coding-system>

  デフォルトのコーディングシステム(fall-back)を coding-system に設定する。

  利用できるコーディングシステム:

  a: auto-select

  c: iso-2022-cn

  j: iso-2022-jp

  k: iso-2022-kr

  ec: euc-china

  ej: euc-japan

  ek: euc-korea

  et: euc-taiwan

  u7: UTF-7

  u8: UTF-8

  l1..9: iso-8859-1..9

  l0: iso-8859-10

  lb,ld,le,lf,lg: iso-8859-11,13,14,15,16

  s: shift-jis

  b: big5

  h: HZ

  r: raw mode

  例:

-Il2: 入力コーディングシステムを iso-8859-2 に設定

-Ks: キーボードコーディングシステムを shift-jis に設定

-Oek: 出力コーディングシステムを euc-korea に設定

-Ab: 全てのコーディングシステムを big5 に設定

コーディングシステム変換 / コードポイント変換:

iso-2022-cn、-jp、-kr は、それぞれ euc-china、-taiwan、euc-japan、euc-korea と相互に変換することができる。shift-jis は iso-2022-jp、euc-japan と同じ内部コードポイントを利用する。

big5 文字列は無視できる程度の不完全さで CNS 11634-1992 に変換できることから、big5 は iso-2022-cn、euc-taiwan とコードポイントを含め相互に変換することができる。ここで参照する iso-2022-cn は CNS シーケンスのみであり、GB シーケンスではない。Note that the iso-2022-cn referred here is not GB sequence, only just CNS one. lv は big5 から GB に直接変換できないことに注意すること。

表示されているコードと内部コードが異なるため、lv がさらにコードポイントの変換(コーディングシステムの変換ではない)を実行するとき、lv の検索機能は正しく機能しない場合がある。lv はこの問題に対して検索パターンの文字列を自動的に変換することで対処しようとするが、この機能は必ずしも完璧ではない。

- -W<number>

  スクリーンの幅
- -H<number>

  スクリーンの高さ
- -E'<editor>' (default 'vi -c %d')

  エディタ名 (デフォルトは'vi -c %d')。"%d"はファイル中の現在の位置の行番号を意味する。
- -q

  delete/insert 行コントロールがあることを指定するこのオプションは delete/insert 行の機能がある MSDOS ANSI ターミナルに設定する。termcap と terminfo バージョンの場合はは自動的に設定される。
- -Ss<seq>

  ANSI の強調シーケンスを seq に設定する (デフォルトは 7)
- -Sr<seq>

  ANSI のリバースシーケンスを seq に設定する (デフォルトは 7)

- -Sb<seq>

  ANSI のブランクシーケンスを seq に設定する（デフォルトは 5）

- -Su<seq>

  ANSI のアンダーラインシーケンスを seq に設定する（デフォルトは 4）

- -Sh<seq>

  ANSIのハイライトシーケンスをseqに設定する（デフォルトは1)このシーケンスは完全なANSIエスケープシーケンスを作成するために"ESC ["と"m"の間に挿入される。

- -T<number>

  ユニコードのコードポイントを2つの地域に分けるThreshhold-codeを設定する地域によって1バイト幅と2 バイト幅のキャラクタが存在します。(デフォルト: 12288, = 0x3000)

- -m

  特定のアジアのキャラクターセットにおいて、Unicode から対応するコードポイントを持つ異なるキャラクターセットに変換する際、iso-8859-*にマップされる iso-8859-*と同じシンボルを持つユニコードコードポイントを強制する。

- -a

  検索パターンのためのキャラクタを調整する（デフォルト）

- -c

  テキスト装飾に ANSI エスケープシーケンスの使用を許可する（色）

- -d、-i

  正規表現検索を無視する(デフォルト)

- -f

  正規表現のための固定文字列を代替する

- -k

  デコーディング中に X0201 Katakana を X0208 に変換する

- -l

  スクリーン再描画の後、各論理行の物理行がカットアンドペーストのために連結されて画面に表示されることを許可する。

- -s

  古いページをスクリーンからスムーズに消すことを強制する

- -u

  いくつかのキャラクタセットを統合する。たとえば、JIS X0208 と C6226. さらに、lv は ISO 646 variants を同一に扱う。たとえば、JIS X0201-Roman と ASCII を含んだ未知のキャラクタセット

- -g

  lgrep モードを有効にする。

- -n

  lgrep の表示結果の各行の先頭に行番号を表示する

- -v

  lgrep で結果を反転させ、マッチしなかった行を表示する

- -z

  HZ の自動検出を有効にする（実行時に C-t コマンドによっても有効になる).

- -+

  全てのオプションを消去する。+c、+d、... +z のように+<option>を使えば、特定のオプションを無効にすることができる。

- -

  以降の引数をファイルネームとみなす

- grep pattern

  grep のような動作をする。(lgrep)

- -V

  lv のバージョンを表示する

- -h

  ヘルプを表示する

設定

オプションはホームディレクトリの.lv ファイル(MSDOS では_lv)で設定することができる。MSDOS の場合に限り、_lv ファイルはカレントの作業ディレクトリに置く必要がある。また、オプションは環境変数 LV に設定することもできる。全ての設定は読み込まれた順に上書きされる。コマンドラインオプションは常に最後に読み込まれる。

コマンドキーバインド

- 0～9

  引数

- g、<

  指定行にジャンプする（デフォルト: ファイルの最上部）

- G、>

  指定行にジャンプする（デフォルト: ファイルの最下部）

- p

  パーセント指定した行にジャンプする (0-100)

- b、C-b

  前のページ

- u、C-u

  前の半ページ

- k、w、C-k、y、C-y、C-p

  前の行

- j、C-j、e、C-e、C-n、CR

  次の行

- d、C-d

  次の半ページ

- f、C-f、C-v、SP

  次のページ

- F

  ファイルの終りまでジャンプし、ファイルにデータが追加されるのを待つ。

- /<string>

  順方向に string 文字列を検索する（正規表現）

- ?<string>

  逆方向に string 文字列を検索する（正規表現）

- n

  順方向に繰り返し検索を行う

- N

  逆方向に繰り返し検索を行う

- C-l

  全ての行を再表示する

- r、C-r

  スクリーンとメモリをリフレッシュする

- R

  カレントファイルを再読み込みする

- :n

  次のファイルを検査する

- :p

  前のファイルを検査する

- t

  入力コーディングシステムを切替える

- T

  入力コーディングシステムを逆方向に切替える

- C-t

  HZ デコーディングモードを切替える

- v

  オプション-E によって定義されたエディタを実行する

- C-g、=

  ファイルの情報を表示する（ファイル名、行数、コーディングシステム）

- V

  lv のバージョンを表示する

- C-z

  サスペンド（シェルまたは command.com を呼び出す）

- q、Q

  終了

- UP/DOWN

  前/次の行

- LEFT/RIGHT

  前/次の半ページ

- PageUp/PageDown

  前/次のページ

検索文字列の入力方法

- C-m、Enter

  現在の文字列を入力する

- C-h、BS、DEL

  1 文字削除する（バックスペース）

- C-u

  現在の文字列をキャンセルし、再度入力する

- C-p

  以前検索した文字列を表示する（履歴）

- C-g

  終了

正規表現

^、$、.、*、+、?、[、^、-、]、¥は特別なキャラクタとして扱われる。¥| 選択肢を指定する¥(, ¥) は構造体をグルーピングする¥1 と¥2 はそれぞれ 1 つか 2 つの列で構成される全てのキャラクターセットにマッチする。相互に重なりあう範囲またはキャラクターセットは保証されない。

関連項目

LV Homepage: http://www.ff.iij4u.or.jp/~nrt/lv/

6.2.5.18 ファイルを表示するためのフィルター (more)

書式

```
more [-dlfpcsu] [-num] [+/ pattern] [+ linenum] [file ...]
```

説明

more は、テキストを一画面ずつ表示するフィルターである。本コマンドは基本的な機能だけを備えている。less(1)は more(1)をエミュレートし、さらに拡張機能を有する。

オプション

コマンドラインオプションを下記に示す。オプションは環境変数 MORE によっても指定される(必ず‘‘-’’(ダッシュ)を前につけること)が、コマンドラインオプションが優先される。

- -num
  スクリーンサイズ(行数)を整数で指定する。
- -d
  more は、"[スペースキーを押すと続き、’q’で終了。]"とユーザーに促し、不適切なキー入力に対しては、ビープ音を鳴らす代わりに"[’h’キーで操作方法]"と表示する。
- -l
  通常 more は、^L(改頁)を特殊文字として扱い、改頁の次の行で停止する。-l オプションは、この機能を抑制する。
- -f
  画面行数の代わりに、論理行をカウントする。(すなわち、長い行が折り返されない。)
- -p
  スクロールしない。その代わりに、全画面消去してからテキストを表示する。
- -c
  スクロールしない。その代わりに、表示されたままの行を消しながら、上端から各画面を表示する。
- -s
  複数の空行を一行にする。
- -u
  下線を付けない。
- +/
  各ファイルが表示される前に検索する文字列を指定する。
- +num
  num 行目から表示する。

コマンド

more の対話的コマンドは、vi(1)をベースにしている。幾つかのコマンドは、10 進数値を前につけることができる。下記の説明においてはその数値は k と表されている。下記文中で、^X は Control-X を表す。

- h または?
  ヘルプ。これらのコマンドのまとめを表示する。もし他のコマンドをすっかり忘れたのなら、これを思い出すと良い。
- SPACE
  k 行先を表示する。デフォルトは現在の画面行数。
- z
  k 行先を表示する。デフォルトは、現在の画面行数。引き数が新たなデフォルトとなる。
- RETURN
  k 行先を表示する。デフォルトは 1。引き数が新たなデフォルトとなる。
- d または^D
  k 行スクロールする。デフォルトは現在のスクロールサイズ、最初は 11。引き数が新たなデフォルトとなる。
- q または Q または INTERRUPT
  終了する。
- s
  k 行先にスキップする。デフォルトは 1。
- f
  k 回画面をスクロールする。デフォルトは 1。ファイルに対してのみ動作する。パイプに対しては動作しない。
- b または^B
  k 回画面をバックスクロールする。デフォルトは 1。
- '
  前回検索を開始した場所に戻る。
- =
  現在の行数を表示する。
- /pattern
  正規表現に k 回目に合致する文字列を検索する。デフォルトは 1。
- n
  前回指定した正規表現に k 回目に合致する文字列を検索する。デフォルトは 1。
- !<cmd>または:!<cmd>
  <cmd>をサブシェルで実行する。
- v

現在の行でエディタを起動する。エディタは、環境変数 VISUAL が定義されていれば VISUAL から、VISUAL が未定義で、EDITOR が定義されていれば EDITOR で指定されているものを起動する。どちらも未定義ならデフォルトとして"vi"を起動する。

- ^L

  画面を再描画する。

- :n

  k 個目のファイルに移動する。デフォルトは 1。

- :p

  k 個前のファイルに移動する。デフォルトは 1。

- :f

  ファイル名と行数を表示する。

- .

  前回のコマンドを繰り返す。

環境変数

more は、下記の環境変数があれば適用する。

- MORE

  more に対する好みのオプションをセットする。

- SHELL

  使用中のシェル。(通常ログイン時にシェルによってセットされる)

- TERM

  ターミナルタイプを指定する。これは more がスクリーン操作に必要とするターミナルの特徴を得るために利用される。

関連項目

vi(1)、less(1)

6.2.5.19 行番号を付けてファイルを出力する (nl)

書式

```
nl [-p] [-b STYLE] [-d CD] [-f STYLE] [-h STYLE] [-i NUMBER] [-l NUMBER] [-n FORMAT]
[-s STRING] [-v NUMBER] [-w NUMBER] [--body-numbering=STYLE] [--footer-numbering=STYLE]
[--header-numbering=STYLE] [--join-blank-lines=NUMBER] [--page-increment=NUMBER]
[--no-renumber] [--number-format=FORMAT] [--number-separator=FORMAT]
[--number-width=NUMBER] [--section-delimiter=CD] [--starting-line-number=NUMBER]
[FILE...]
```

nl [--help] [--version]

説明

nl は指定された FILE それぞれを標準出力に書く。その際行番号を一部または全部の行に追加する。FILE が一つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

nl は入力を論理ページ(logical page)に分解する。デフォルトでは、行番号は各々の論理ページの先頭で 1 にリセットされる。nl は入力されたファイルすべてをまとめてひとつの文書とみなす。したがってファイルの切れ目でも行番号や論理ページはリセットされない。

オプション

- -b STYLE、--body-numbering=STYLE

  各論理ページの本文セクションにおける行の番号付けの方式を選択する。行に番号がつかないと、カレントの行番号は増加しないが、行番号の区切る文字は行の前に置かれる。形式は以下の通り。

  a　全ての行に番号をふる。

  t　空行でない行のみに番号をふる(本文セクションのデフォルト)

  n　行番号をふらない(ヘッダ・フッタセクションのデフォルト)

  pREGEXP　REGEXP にマッチした行だけに番号をふる。
- -d CD、--section-delimiter=CD

  セクション区切りとなる二文字を CD にする(デフォルトは 'e:')。C だけが与えられた場合は、2 文字めは ':' のままとなる。('¥' 等のメタキャラクタを指定する場合には、シェルの展開から守るためにクォートするか、バックスラッシュを一つ余計に付けるのを忘れないように。)
- -f STYLE、--footer-numbering=STYLE

  --body-numbering を見よ。
- -h STYLE、--header-numbering=STYLE

  --body-numbering を見よ。
- -i NUMBER、--page-increment=NUMBER

  行番号の増分を NUMBER にする(デフォルトは 1)。
- -l NUMBER、--join-blank-lines=NUMBER

  NUMBER だけ連続した空行を、番号付けの際に 1 論理行とみなし、最後の空行にのみ番号を付ける(NUMBER のデフォルトは 1)。空行の連続が number 以下の場合には、番号付けを行わない。空行とはスペースやタブも含め、文字を全く含まない行のこと。
- -n FORMAT、--number-format=FORMAT

行番号の出力フォーマットを指定する(デフォルトは 'rn'):

ln　　左詰めにし、先頭の 0 埋めをしない。

rn　　右詰めにし、先頭の 0 埋めをしない。

rz　　右詰めにし、先頭は 0 で埋める。

- -p、--no-renumber

  論理ページの先頭で行番号をリセットしない。

- -s STRING、--number-separator=FORMAT

  出力ファイルの行番号とテキスト行の間の区切りに STRING を用いる(デフォルトは<TAB>)。

- -v NUMBER、--starting-line-number=NUMBER

  それぞれの論理ページの行番号を NUMBER から始める(デフォルトは 1)。

- -w NUMBER、--number-width=NUMBER

  行番号表示に用いる文字数を NUMBER にする(デフォルトは 6)。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

論理ページ

論理ページは三つのセクションからなる。ヘッダ(header)、本文(body)、フッタ(footer)である。セクションは空であってもよい。また、それぞれを別な形式で番号付けすることもできる。

入力ファイル中に以下のデリミタ文字列(delimiter string)だけからなる行があると、論理ページのセクションの始まりとみなされる。

- ¥:¥:¥:

  ヘッダの先頭

- ¥:¥:

  本文の先頭

- ¥:

  フッタの先頭

これらの文字列を構成する二文字は、オプションによって '¥' と ':' 以外にも変更できる。しかしパターンとそれぞれの文字列の長さは変更できない。セクションのデリミタ文字列は出力では空行となる。入力ファイルのうち、最初のセクションデリミタ文字列の前に来るテキストは本文セクションとみなされる。したがって nl は、セクションのデリミタ文字列を含まないファイルを単一の本文セクションからなるものとみなす。

6.2.5.20 ファイルを 8 進数(または他の形式)で出力する (od)

書式

```
od [-v] [-j BYTES] [-s [LENGTH]] [-t TYPE] [-w WIDTH] [-A RADIX] [-NBYTES]
[--address-radix=RADIX] [--format=TYPE] [--output-duplicates] [--read-bytes=BYTES]
[--skip-bytes=BYTES] [--strings[=LENGTH]] [--width[=WIDTH]] [FILE...]

od [--help] [--version]
```

非 POSIX オプション:

```
od [-abcdfhilox] [FILE...]
od -C [--traditional] [FILE] [[+]OFFSET [[+]LABEL]]
```

説明

od は FILE それぞれの内容を明確な形式で標準出力に書き出す。FILE が 1 つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

出力のそれぞれの行では、まず最も左のカラムに入力ファイルでのオフセットが書かれ、ファイルのデータのグループが続く。デフォルトでは od はオフセットを 8 進数で、ファイルのデータのグループは 2 バイトずつ 8 進数で表す。

オプション

- -j BYTES、--skip-bytes=BYTES

  整形・表示する前に入力ファイルの BYTES 分をスキップする。BYTES が '0x' または '0X' で始まる場合は 16 進数と解釈される。これ以外で先頭が '0' の場合は 8 進数、それ以外は 10 進数と解釈される。'b' を後置すると BYTES は 512 倍、'k' なら 1024 倍、'm' なら 1048576 倍される。
- -s [LENGTH]、--strings[=LENGTH]

  通常の出力の代わりに、入力の文字列定数(string constants)のみを出力する。文字列定数とは、最低 LENGTH の連続した ASCII グラフィック(フォーマット)文字で、null(ゼロ)バイトによって終端されているものである。LENGTH が省略された場合のデフォルトは 3。短縮形式-s は推奨されない。
- -t TYPE、--format=TYPE

  ファイルデータの出力フォーマットを選択する。TYPE は文字列で、以下のタイプ指定文字からなる。1 つの TYPE 文字列中に複数のタイプ指定文字を書いたり、このオプションを複数回用いると、od は指定文字 1 つにつき 1 行の出力行を書き出す。行の出力順は指定文字の順番になる。

  'z' を後置すると、どんな形式指定に対しても、形式指定によって生成された行に印刷可能文

字での ASCII 文字表示を追加する。

a　文字の名前
c　ASCII 文字かバックスラッシュ付きのエスケープ文字
d　符号付き 10 進数
f　浮動小数点数
o　8 進数
u　符号無し 10 進数
x　16 進数

‘a’ 形式は空白文字に ‘sp’ ,改行文字に ‘nl’ ,null(ゼロ)バイトに ‘nul’ 等を出力する。‘c’ ではこれらはそれぞれ ‘’ , ‘¥n’ , ‘¥0’ となる。
‘a’ と ‘c’ 以外のタイプについては、入力データから何バイト分を使うかを指定できる。これには形式表示文字の後に 10 進の整数を後置する。または C コンパイラの組み込みデータ型によってもサイズを指定できる。以下のうちのどれかをタイプ指定の後に指定する。
整数(‘d’ , ‘o’ , ‘u’ , ‘x’)については以下のどれか:

C　char
S　short
I　int
L　long

浮動小数点数(‘f’)については以下のどれか:

F　float
D　double
L　long double

- -v、--output-duplicates

  直前と同じ内容を持つ行も表示する。デフォルトでは、連続する出力行が同じ内容である場合には、od は最初の行のみを表示し、続く行には省略した旨を伝えるアスタリスク(*)のみを置く。

- -w[WIDTH]、--width[=WIDTH]

  出力ファイルの 1 行あたり WIDTH バイトの入力を表示する。これは出力タイプに指定した各サイズの公倍数でなければならない。WIDTH が省略された場合のデフォルトは 32。このオプションが指定されなかった場合のデフォルトは 16。短縮形式-w は推奨されない。

- -A RADIX、--address-radix=RADIX

  表示されるオフセットの基数を選択する。RADIX として指定できるのは以下のうちのどれか:

  d　10 進数

  o　8 進数(デフォルト)

  x　16 進数

  n　なし(オフセットを表示しない)

- -N BYTES、--read-bytes=BYTES

  入力ファイルのうち BYTES に達するまでしか表示しない。BYTES で前置・後置される文字は -j オプションと同様に解釈される。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

非 POSIX オプション

- -a

  文字の名前を出力する。-ta と等価。

- -b

  8 進でバイトを出力する。-toC と等価。

- -c

  ASCII 文字またはバックスラッシュ付きのエスケープ文字として出力する。-tc と等価。

- -d

  符号無し 10 進 short として出力する。-tu2 と等価。

- -f

  float として出力する。-tfF と等価。

- -h

  16 進 short として出力する。-tx2 と等価。

- -I

  10 進 short として出力する。-td2 と等価。

- -l

  10 進 long として出力する。-td4 と等価。

- -o

  8 進 short として出力する。-to2 と等価。

- -x

16 進 short として出力する。-tx2 と等価。

- -C、--traditional

  POSIX 以前の、オプション以外の引き数のうち、古いバージョンの od が受け付けたものを認識する。

たとえば以下の書式:

od --traditional [FILE] [[+]OFFSET[.][b] [[+]LABEL[.][b]]]

を使うと、最大 1 つのファイルと、オフセットおよび擬似スタートアドレス LABEL を指定できる。デフォルトでは OFFSET は 8 進数と解釈され、整形・出力する前にスキップする入力ファイルのバイト数を示す。小数点を付加すると、OFFSET は 10 進数とみなされる。10 進の指定がなく、offset が '0x' または '0X' で始まる場合は 16 進数と解釈される。'b' が後置されると offset を 512 倍したバイト分がスキップされる。LABEL 引き数も OFFSET のように解釈されるが、これはスタート地点の擬似アドレスを指定する。擬似アドレスは通常のアドレスの後に括弧付きで表示される。

6.2.5.21 ファイルを行単位でマージする (paste)

書式

paste [-s] [-d DELIM-LIST] [--delimiters=DELIM-LIST] [--serial] [FILE...]

paste [--help] [--version]

説明

paste は与えられたそれぞれの FILE から、同じ行番号の行を連結して標準出力に書き出す。区切り文字には<TAB>が用いられる。FILE が一つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

オプション

- -d DELIM-LIST、delimiters=DELIM-LIST

  マージするファイル間のセパレータに、<TAB>ではなく DELIM-LIST を順番に用いる。DELIM-LIST の文字を使い果たしたら、再び最初から用いはじめる。

- -s、--serial

  各ファイルから 1 行づつ読み込むのではなく、ファイル単位で 1 行にまとめていく。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.22 印刷用にファイルのページづけ・段組みを行う (pr)

書式

```
pr [+FIRST_PAGE[:LAST_PAGE]] [-COLUMN] [-acdfmrtvFJT] [-e[INTABCHAR[IN-TABWIDTH]]
[-h HEADER] [-i[OUT-TABCHAR[OUT-TABWIDTH]] [-l PAGE_LENGTH]
[-n[NUMBER-SEPARATOR[DIGITS]] [-o MARGIN] [-s[SEP-CHAR]] [-w PAGE_WIDTH]
[-D DATEFORMAT] [-N LINE_NUMBER] [-S[SEP-STRING]] [-W PAGE_WIDTH] [--across]
[--columns=COLUMN] [--date=DATEFORMAT] [--double--space]
[--expand-tabs[=IN-TABCHAR[IN-TABWIDTH]] [--form-feed] [--header HEADER]
[--indent=MARGIN] [--length PAGE_LENGTH] [--merge]
[--number-lines[=NUMBER-SEPARATOR[DIGITS]] [--no-file-warnings]
[--output-tabs[=OUT-TABCHAR[OUT-TABWIDTH]] [--omit-header]
[--pages=FIRST_PAGE[:LAST_PAGE]] [--page_width=PAGE_WIDTH] [--separator[=SEP-CHAR]]
[--sep-string[=SEP-STRING]]         [--show-control-chars]         [--show-nonprinting]
[--width=PAGE_WIDTH]
[FILE...]

pr [--help] [--version]
```

説明

pr は FILE の内容にページを付けて標準出力に書き出す。オプションで指定すれば段組みして出力することもできる。FILE が 1 つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

デフォルトでは、それぞれのページに5行のヘッダがつく。ヘッダは2行の空行、日付・ファイル名・ページ番号の行、2 行の空行からなる。各ページの末尾には 5 行の空行からなるフッタも出力される。

デフォルトでは PAGE_LENGTH は 66 行で、ヘッダのテキスト行は PAGE_WIDTH 桁(デフォルトは 72 桁)を幅いっぱいに用い、'yy-mm-dd HH:MM StringPage nnnn' という形式で出力される。ここで String はまん中におかれる文字列である。

入力にフォームフィードがあると、そこで出力は改頁される。フォームフィードが続くと空のページができる。

段は、おのおの等しい幅を持ち、デフォルトではスペースで区切られる。-J オプションが指定されない限り、行は常に PAGE_WIDTH で切り捨てられる。1 段の出力では、行の切り捨てはデフォルトでは行われない(切り捨てを行うには-W オプションを使う)。

オプション

- +FIRST_PAGE[:LAST_PAGE]
- --pages=FIRST_PAGE[:LAST_PAGE]

  FIRST_PAGE から LAST_PAGE までを出力する。LAST_PAGE を省略するとファイル末尾までを

出力する。スキップするページ数を評価するとき、入力ファイルにあるフォームフィードは改ページとみなされる。+FIRST_PAGE があってもなくても、出力のページ番号と行番号は同じで、入力の最初のページからカウントが開始される(出力ページの最初からではない)。しかしページ番号の振り方は-N によって変更することもできる。

- -COLUMN
- --columns=COLUMN

  FILE それぞれが COLUMN 段の出力になる(デフォルトは 1)。段の幅は PAGE_WIDTH から自動的に計算され、それぞれのページにおける各段の行数は調整される。段に入りきらない行は切り捨てられる。--columns は-m と同時には指定できない。
- -a、--across

  それぞれの FILE で、段組みの行の進行方向を縦でなく横にする。COLUMN は 2 以上でなければならない。段に収まらない行の余計な分は切り捨てられる。
- -c、--show-control-chars
- -d、--double-space

  行間に空行を挿入する(ダブルスペース表示)。
- -e[IN-TABCHAR[IN-TABWIDTH]]
- --expand-tabs[=IN-TABCHAR[IN-TABWIDTH]],

  入力のタブをスペースに展開する。IN-TABCHAR は入力のタブ文字を指定する(省略可:デフォルトは<TAB>)。IN-TABWIDTH はタブの幅を指定する(省略可:デフォルトは 8)。
- -f、-F、--form-feed

  改ページに複数の改行ではなくフォームフィード(^L)を使用し、ヘッダは 3 行形式にする(最初の空行 2 つとフッタは省略される)。1 ページあたり 66 行というデフォルトの設定は変更されないが、1 ページあたりのテキスト行はデフォルトの 56 行から 63 行に変更される。
- -h HEADER、--header=HEADER

  ヘッダに表示される FILE の名前を文字列 HEADER に置き換える。‘yy-mm-dd HH:MM HEADER Page nnnn’が PAGE_WIDTH より長くなると、左側で行の切り捨て(‘*’マークされる)が行われる。‘-h""’とするとヘッダ行は空になる。
- -i[OUT-TABCHAR[OUT-TABWIDTH]]
- --output-tabs[=OUT-TABCHAR[OUT-TABWIDTH]]

  出力のスペースをタブに圧縮する。OUT-TABCHAR は出力のタブ文字を指定する(省略可:デフォルトは<TAB>)。OUT-TABWIDTH はタブの幅を指定する(省略可:デフォルトは 8)。
- -J、--join-lines

  行の内容すべてをマージする。段組みオプション-COLUMN,‘-a-COLUMN’、-m とともに指定すると、-W や-w の行切り捨てを無効にする。段の揃えは行わない。段の区切り文字は<TAB>に変更される。ただし-S による上書きは可能である。
- -l PAGE_LENGTH、--length=PAGE_LENGTH

(ヘッダとフッタを含めた)ページ長を PAGE_LENGTH 行に設定する。デフォルトは 66 行。PAGE_LENGTH が 10 行以下の場合(および-F も指定された場合は 3 行以下の場合)は、page-length が 10 以下の場合には、-T オプションが指定されたかのように、ヘッダとフッタが省略され、入力ファイルのフォームフィードは無視される。

- -m、--merge

  全てのファイルを並べて出力する。1 ファイルが 1 段となる。行の長さが段に収まらない場合、はみ出た分は切り捨てられる。FILE に空のページが(フォームフィードによって)存在すると、空の段が出力される。ただし SEPARATOR によるマークはされる。(しかし全ての段が空となるページでは、区切り文字も行番号も表示されない。)デフォルトのヘッダは 'yy-mm-dd HH:MM <blanks> Page nnnn' となるが、-h HEADER を同時に指定すれば、まん中の部分を記述するようにできる。-S[SEP-CHAR]も同時に指定できる。

- -n[NUMBER-SEPARATOR[DIGITS]]

- --number-lines[=NUMBER-SEPARATOR[DIGITS]]

  各段の前に行番号をふる。複数ファイルの並列表示(-m)の場合は、行の先頭だけに番号をふる。行番号の開始は、デフォルトでは入力ファイルの最初の行である(印刷された最初の行ではない。-N と-PAGES を見よ)。

  NUMBER-SEPARATOR は行番号とテキストの間に入る文字を指定する。デフォルトはスペース。桁数が変わらない場合は<TAB>に圧縮される。DIGITS は行番号の桁数を指定する(省略可:デフォルトは 5)。

- -o MARGIN、--indent=MARGIN

  各行を MARGINP 文字分のスペースでインデントする(デフォルトは 0)。つまり左マージンを設定する。総ページ幅は MARGIN+PAGE_WIDTH になる。行番号つきの段組み出力では溢れることがあるかもしれない。

- -r、--no-file-warnings

  コマンドラインで与えられた FILE がオープンできない場合でも、警告メッセージを表示しない。(しかしファイルのオープンに失敗した場合に 0 以外の終了ステータスを返す動作は変らない。)

- -s[SEP-CHAR]、--separator[=SEP-CHAR]

  段組のセパレータを SEP-CHAR 文字にする。SEP-CHAR のデフォルトは、-s と同時に-w を指定した場合は空文字列、それ以外の場合は<TAB>。-w と組み合わせた場合には、段の行切り捨てが行われないという副作用がある。

- -S[SEP-STRING]、--sep-string[=SEP-STRING]

  段組みのセパレータを SEP-STRING にする(デフォルトは空文字列)。短縮形式-S は推奨されない。

- -t、--omit-header

  各ページのヘッダ(とフッタ)を印字しない。また改ページ処理も行わない(ページ末に空行やフ

ォームフィードを入れない)。ページ構造は生成されないが、入力ファイルのフォームフィードは残る(つまりあらかじめ定義されたページ区切りは変更されない)。-t や-T は他のオプションと組み合わせると便利である。例えば '-t -e4' は入力の<TAB>を 4 つのスペースに変換する以外は何も変更しない。-t は-h を無効にする。

- -v、--show-nonprinting

  印字できない文字を、バックスラッシュ表記の 8 進数で出力する。

- -w PAGE_WIDTH、--width=PAGE_WIDTH

  複数段の出力に限り、ページは場を PAGE_WIDTH 文字に設定し(デフォルトは 72)、行の余分を切り捨てる。

- -D DATEFORMAT、--date=DATEFORMAT

  DATEFORMAT をヘッダの日付書式として使用する。または日付を指定するために使用する。詳細は date(1)を見よ。

- -N LINE_NUMBER、--first-line-number=LINE_NUMBER

  印刷された最初のページの最初の行を LINE_NUMBER 行とし、そこから行カウントを開始する。

- -S [SEP-STRING]、--sep-string[=SEP-STRING]

  出力の段組みの区切りを文字列 SEP-STRING にする。

- -T

  ヘッダ(とフッタ)を出力しない。さらに入力にあるフォームフィードをすべて無効にする。

- -W PAGE_WIDTH、--page_width=PAGE_WIDTH

  段組みが 1 段の場合でも複数の場合でも、ページの幅を PAGE_WIDTH 文字にして(デフォルトは 72)、行の余分を切り捨てる。-W の指定も段組みのオプションも指定されなければ、行の余分の切り捨ては行われない(ヘッダは例外で、常に切り捨てが行われる)。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

POSIX 向けの注意

大文字 1 文字のオプションは小文字のオプションより優先される。しかし後の方に再定義するほうが POSIX には準拠している。また、1 文字のオプションへの引数は、POSIX の規定ではオプションの文字と分離できない(例えば '-s a' は '-sa' と書かなければならない)。

6.2.5.23 整列済み索引を作成する (ptx)

書式

ptx [OPTION...] [FILE...]

ptx [-G] [--traditional] [OPTION...] [IN-FILE [OUT-FILE]]

オプション

[-frAORT] [-b FILE] [-g NUMBER] [-i FILE] [-o FILE] [-w NUMBER] [-F STRING] [-M STRING] [-S REGEXP] [-W REGEXP] [--auto-reference] [--break-file=FILE] [--flac-truncation=STRING] [--format={nroff,tex}] [--gap-size=NUMBER] [--ignore-case] [--ignore-file=FILE] [--macro-name=STRING] [--only-file=FILE] [--references] [--right-side-refs] [--sentence-regexp=REGEXP] [--word-regexp=REGEXP] [FILE...]

ptx [-C] [--copyright] [--help] [--version]

説明

GNU 拡張された ptx(デフォルト)は、与えられた FILE それぞれの整列済み索引(permutated index)を標準出力に書く。FILE が一つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。結果は結合されるが、各 FILE はそれぞれ独自のコンテクストを持ち、自動参照を使うときには別々に参照される。[訳注:参照(reference)とは、キーワードの現れるファイル名と行数の表示。] --traditional モードを使うと、ptx は入力を IN-FILE から読み、OUT-FILE に書く。後者が省略されると標準出力に書く。

デフォルトの出力フォーマットは、キーワードをセンターに、そしてコンテクストがあれば左または右に書く。--traditional モードでは--format=nroff が用いられる。

オプション

- -b FILE、--break-file=FILE

  単語に含むことのできない文字を FILE から取得する(このファイルは break file と呼ばれる)。デフォルトのモードでは、FILE にある全ての文字が(改行文字も含めて)考慮される。過去互換モードでは、空白・タブ・改行の各文字は、常に FILE からは捨てられる。

- -f、--ignore-case

  文字列をソートするとき英大文字小文字を無視する。

- -g NUMBER、--gap-size=NUMBER

  フィールド間の空白の最低数を NUMBER に設定する(デフォルトは 3)。

- -i FILE、--ignore-file=FILE

  無視するキーワードのリストを FILE から取得する(このファイルは ignore file と呼ばれる。デフォルトは/usr/local/lib/eign)。各行には単語を一つだけ指定する。ignore file にある単語は

only file にある単語を上書きする。

- -o FILE、--only-file=FILE

  キーワードのリストを FILE から取得する(このファイルは onlyfile と呼ばれる)。インデックスを生成するとき、このリストにない単語は無視する。各行には単語を一つだけ指定する。

- -r、--references

  各行の前にコンテクストを指示する文字列(行頭の単語)を付ける。ptx は参照をコンテクストから削除しようとし、コンテクストが改行で終わる場合にはこれは常に成功する。このオプションを-S と共に用いたり(これはデフォルト)、--traditional モードを用いると、参照は常にコンテクストから削除される。

- -w NUMBER、--width=NUMBER

  出力行を NUMBER 桁を越えないように切り捨てる。--right-side-refs の分は含まれないので注意。これを用いた場合は指定桁を越えることがある。

- -A、--auto-reference

  各行の前にファイル名(標準入力から読み込んだ場合は空文字列)、行番号、コロンを出力する。--references より優先する。

- -F STRING、--flac-truncation=STRING

  --width によって行を切り捨てるときや、コンテクストが行区切りを越えて続く場合に STRING(デフォルトは '¥')を出力する。

  STRING 中では、(C プログラムで使うような)バックスラッシュを用いたエスケープシーケンスの多くも認識され、適切な文字に変換される。

- -M STRING、--macro-name=STRING

  nroff や TeX 形式で出力するときに、STRING をマクロ名に使う(デフォルトは '.xx')。

- -O、--format=nroff

  出力を nroff 形式にする。印字できない文字はスペースに置換され、クォート文字は二重にして正しく処理できるようにする。各行のフォーマットは以下の通り:

  .xx "TAIL" "BEFORE" "KEYWORD_AND_AFTER" "HEAD" "REF"

- -R、--right-side-refs

  --references と似ているが、参照を右に出力する。

- -S REGEXP、--sentence-regexp=REGEXP

  REGEXP を行末または文末の評価に用いる。GNU モードで--references オプションが指定されていない場合のデフォルトは以下:

  [.?!][]¥"')]*¥¥($¥¥|¥t¥¥|¥¥)[¥t¥n]*

  --traditional モードや、GNU モードで--references オプションが指定されている場合のデフォルトは以下:

  ¥n

  -F と同様、バックスラッシュを用いたシーケンスも認識・変換される。

- -T、--format=tex

  出力を TeX 形式にする。印字できない文字はスペースに置換し、いくつかの特殊文字('$' 、'%' 、'&' 、'#' 、'_' など)をバックスラッシュでプロテクトする。バックスラッシュは '¥backslash{}' に、アクセント記号は '^¥{}' に、チルダは '~¥{}' に置換し、その他の音韻記号も可能な限り最も適切な TeX シーケンスに変換する。各行のフォーマットは以下の通り:

  ¥xx {TAIL}{BEFORE}{KEYWORD}{AFTER}{HEAD}{REF}

- -W REGEXP、--word-regexp=REGEXP

  REGEXP にマッチする単語をキーワードにする。--break-file に指定されている単語も出力する。GNU モードでのデフォルトは:

  ¥w+

  --traditional モードでのデフォルトは

  [^ ¥t¥n]+

  REGEXP が空の場合はデフォルトを用いる。-F と同じく、バックスラッシュを用いたシーケンスも認識・変換される。

- -C、--copyright

  標準出力に短い copyright メッセージを出力して正常終了する。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

移植性

--traditional を指定すると、System V の ptx と互換になる。GNU ptx は行幅をよりうまく使えるのだが、この点も System V の出力に時々現れる異常をまねしようと試みる。--traditioanl モードとの他の相違点は以下の通り:

- 説明にあるように、引数の取り方が変わる。
- 指定できるオプションが-b、-f、-g、-i、-o、-r、-t、-w だけになる。
- デフォルトの出力形式が--format=nroff になる。
- --width で行切り捨てを行うとき、参照の幅を考慮しなくなる。
- 8 ビット文字とチルダ('~')をはねる。いくつかの制御文字もはねる。
- 入力行の 200 文字以降を黙って切り捨てる。
- ignore file と only file の両方を同時に指定できない。
- オプションに記述したように、いくつかのオプションのデフォルトが変わる。

6.2.5.24 各行ごとに入力された文字を逆に並べたものを出力する (rev)

書式

```
rev [file]
```

説明

rev は、file が指定されていればそのファイルを、指定されていなければ標準入力を読み込み、各行ごとに文字の並びを逆にして出力する。

6.2.5.25 ファイルを繰り返し上書きする (shred)

書式

```
shred [-ITERS] [-fuvxz] [-n ITERS] ["-s SIZE] [--force] [--iterations=ITER] [--size=SIZE]
[--remove]
[--verbose] [--exact] [--zero] FILE[...]

shred [--help] [--version]
```

説明

shred は指定されたファイル FILE を特別なパターンで繰り返し上書きし、データの復旧がより困難になるようにする。FILE が '-' の場合、入力ファイルとして標準入力が使われる。

shred はファイルへの実際の書き込み操作が(訳註:書き込み操作を行ったデバイスと)同じ場所で起ることを仮定して動作するが、(トランザクション管理ファイルシステム・分散ファイルシステム・リモートファイルシステムでは)書き込みが同じ場所で起らない可能性もあるので注意すること。また (RAID が使用されている場合)あるデバイス(/dev/hda など)で shred が使用されると、RAID はこの操作を他のデバイスに対しても同じように行う。

オプション

- -f、--force

  可能であれば、パーミッションを無視する。
- -ITERS、-n ITERS、--iterations=ITERS

  ITERS 回繰り返し上書きする。(デフォルト:25)
- -s SIZE、--size=SIZE

  SIZE バイトのみを切れ切れにする。サイズには乗数の文字を使うことができる(下記参照)。
- -u、--remove

shred を実行した後、切り詰めて(truncate)アンリンクする。

- -v、--verbose

  進捗状況のメッセージを表示する。

- -x、--exact

  全ブロック数以上にファイルサイズを大きくしない。

- -z、--zero

  shred を実行した後、NUL で上書きする。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

乗数

サイズ数値の後には乗数を指定するサイズ文字と、通常のバイトを選択する B または 10 進の「商業用」バイトを選択する D を続けてもよい。例えば ‘1KB’ は ‘1024’ で ‘1KD’ は ‘1000’ である。b(512 バイト、c(1 バイト)、w(これは使用すべきでない-SystemV では 2、4.2BSD では 4 を意味する)は例外であり、B や D を続けることはできない。

- k　キロ:通常のバイトなら 2^10=1024、10 進のバイトなら 10^3=1000
- M　メガ:2^20=1,048,576 または 10^6=1,000,000
- G　ギガ:2^30=1,073,741,824 または 10^9=1,000,000,000
- T　テラ:2^40=1,099,511,627,776 または 10^12=1,000,000,000,000
- P　ペタ:2^50=1,125,899,906,842,624 または 10^15=1,000,000,000,000,000
- E　エクサ:2^60=1,152,921,504,606,846,976 または 10^18=1,000,000,000,000,000,000
- Z　ゼタ:2^70=1,180,591,620,717,411,303,424 または 10^21=1,000,000,000,000,000,000,000
- Y　ヨタ:2^80=1,208,925,819,614,629,174,706,176 または
  10^24=1,000,000,000,000,000,000,000,000

6.2.5.26 テキストファイルをソートする (sort)

書式

```
sort [-cm] [-bdfginruzM] [+POS1[-POS2] [-o OUTFILE] [-t SEPARATOR] [-k POS1[,POS2]
[-K POS1[,POS2] [-S  SIZE] [-T  TEMPDIR] [--reverse] [FILE...]

sort [--help] [--version]
```

説明

sort は与えられた各 FILE をソート・マージ・比較する。結果は結合されて標準出力に書き出される。FILE が一つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

GNU sort は(他の GNU ユーティリティと同じく)入力ファイルの行の長さや、行あたりのバイト数に制限はない。また入力ファイルの最後が改行でなければ、sort は黙って改行を一つ追加する。

エラーが起こると、sort はステータス 2 で終了する。

環境変数 TMPDIR が設定されていると、sort は一時ファイルの置き場所として、デフォルトの/tmp の代わりにそのディレクトリを用いる。オプション-T TEMPDIR でも一時ファイルを置くディレクトリを指定できる。オプションは環境変数より優先される。

-k や+オプションでのソートフィールドの場所指定は、F.C という形式で行う。ここで F は用いるフィールドの番号で、C は先頭の文字の番号を、そのフィールドの先頭(+POS の場合)あるいは直前のフィールドの末尾(-POS の場合)から数えた数字である。.C が省略された場合は、フィールドの先頭文字になる。-b オプションが指定されると、フィールド指定の.C の部分は、そのフィールドの空白でない最初の文字(+POS の場合)あるいは直前のフィールドの末尾以降の空白でない最初の文字(-POS の場合)から数えられる。

ソートキーのオプションには、出力順序オプション(output orderingoption)を含めることもでき、その場合はグローバルな順序オプションはそのフィールドには用いられない。-b オプションはフィールド指定の +POS と-POS のどちらか片方にも、あるいは両方にも、独立に効力を及ぼすことができる。-b がグローバルなオプションから継承されたものである場合は、両方に影響する。キーは複数のフィールドにまたがってもかまわない。

オプション

動作モード

sort のデフォルトの動作はソート動作である。これは以下のオプションによって変更できる。

- -c

  与えられたファイルがすでにソートされているかどうかをチェックする。ソートされていないものがあった場合は、エラーメッセージを表示してステータス 1 で終了する。それ以外の場合は正常終了する。

- -m

  与えられたファイル群をまとめてソートしてマージする。入力ファイルは事前にそれぞれソートされていなければならない。マージ動作ではなく、ソート動作を複数ファイルにまとめて行うこともできる。マージ動作が提供されているのは、(これでよい場合は)こちらの方が高速だから

である。

行のペアは以下のように比較される。キーフィールドが指定されている場合は、sort は順序オプションによってコマンドラインから指定された順に、それぞれの行からの各フィールドを比較する。比較動作は違いを発見するか全てのフィールドを尽くすまで継続される。

グローバルオプションの '-bdfinrM' のいずれかが指定されており、キーフィールドが一つも指定されていない場合は、sort は行全体をグローバルオプションに従って比較する。

最後に、全てのキーが等しい(あるいは順序オプションがまったく指定されなかった)場合には、sort は最終手段として、行を LC_COLLATE ではなくマシンの照合順序にしたがって 1 バイトずつ比較する。この最終手段の比較は-r オプションに従う。-s オプション(stable オプション)は、この最終手段比較を無効にし、全てのフィールドの比較結果が同じだった行の順序を、入力の順序のままにする。フィールドやグローバルオプションがまったく指定されなかった場合は、-s オプションは無視される。

出力順序オプション

以下のオプションは、sort が行を出力する順序に影響する。これらはグローバルにも、あるいは特定のキーフィールドの一部としても指定できる。キーフィールドが指定されなければ、グローバルオプションで行全体が比較される。キーフィールド指定がされた場合、順序オプションが特に指定されなかったフィールドには、グローバルオプションが継承される。

- -b

  各行の比較の際に、行頭の空白を無視する。
- -d

  「電話帳」順でソートする。アルファベット、数字、空白以外のキャラクタをすべて無視してソートする。
- -f

  ソートの際に、小文字を対応する大文字と同じに扱う。例えば 'b' は 'B' と同じとみなされる。
- -g

  strtod(3) を用いて数値に変換した後、その数値順にソートする。これを用いると、浮動小数点の理工学的な記法('1.0e-34' や '10e100' など)を扱うことができる。このオプションは、他に手段がない場合に限って用いること。これは-n よりずっと遅いし、有効桁数が多すぎると、丸めたかたちで比較されてしまう。さらに、倍精度浮動小数点の範囲で扱えない数値は 0 であるかのように処理される。またオーバーフロー・アンダーフロー・変換エラーは報告されない。
- -i

  印字可能でない文字を無視する。
- -n

  数値順に評価する。行先頭の文字列(空白が前置されていても良い)を数値文字列として比較

する。数値文字列は、先頭の-符号(なくても良い)、0 桁以上の数字、そして小数点と 0 桁以上の数字(なくても良い)からなる。

sort は浮動小数点表記文字列を、あまり普通でない方法で比較する。まず各文字を C の double 方に変換してからそれらの値を比較するのではなく、2 つの文字の基数点(radix point)を揃えて、1 文字づつ比較するのである。この方法の利点は速度である。実際この方法は、2 つのそれぞれの文字列を double に変換してから比較するよりずっと速い(あるいは integer 変換に比べてすらずっと速い)。また精度の問題も生じない。多くのシステムでは、文字列を double に変換してから比較する方法では、精度は 16 桁に制限されてしまう。

先頭の '+' や指数表記は認識できない。このような文字列を数値的に比較するには-g を用いること。

- -r、--reverse

  比較の結果を逆順にする。より大きなキー値を持つ行が、より早く現われるようになる。

- -M

  行頭の空白文字をすべて無視して最初に現われた 3 文字を、月の名称の省略形とみなして 'JAN' < 'FEB' <...< 'DEC' の順でソートする。小文字は大文字と同じに扱われる。月の名称にない文字列は、より低位であるとみなされる。

その他のオプション

- +POS1[-POS2]

  ソートフィールド指定の obosolete な古い形式。行の POS1 から POS2 の直前までのフィールドを指定する。POS2 は含まない。POS2 が省略されたら行末まで。フィールドと文字位置はそれぞれ 0 から数えはじめる。

- -k POS1[,POS2]

  -K POS1[,POS2]ソートフィールド指定の POSIX 形式。今後はこちらが推奨される。行の POS1 から POS2 までのフィールドを指定する。POS2 を含む。POS2 が省略されたら行末まで。フィールドと文字位置はそれぞれ 0 から数えはじめる。

- -o OUTFILE

  出力先を標準出力から OUTFILE に変更する。OUTFILE が入力ファイルのどれかひとつだった場合、sort はその入力ファイルを一時ファイルにコピーしてから、ソートと OUTFILE への出力を行う。

- -t SEPARATOR

  各行からソートキーを検索する際、文字 SEPARATOR をフィールドのセパレーターにする。デフォルトでは、フィールドは空白以外の文字と空白文字の間の空文字列(emptystring)によって分離される。例えば入力行として 'foobar' が与えられた場合、sort はこの行をフィールド 'foo' と 'bar' に分離する。フィールドセパレーターは、その前後のフィールドには含まれないものとされる。

- -u

  デフォルトの動作と-m オプションの動作では、等しいとされた行のうちの最初のものだけを表示する。-c オプションの動作では、連続した行で等しいものがないかどうかをチェックする。

- -z

  入力における行の末尾が、<LF>(ラインフィード)ではなく<NUL>(ゼロバイト文字)で終了するとみなす。このオプションは 'perl -0' や 'find -print0' や 'xargs -0' などと組み合わせて使うと便利で、これらは任意のパス名を扱う際に信頼性を上げる効果を持つ(ラインフィード文字が含まれるパス名も正しく扱える)。

- -S SIZE

  SIZE KB のバッファを使う。単位指定文字(後述)を用いれば、単位を変更できる。

- -T TEMPDIR

  TEMPDIR を一時ファイルを置くディレクトリにする。このオプションは環境変数 TMPDIR より優先される。-T オプションが複数回指定されると、それぞれのディレクトリが用いられ、巨大なソートやマージの際には性能が上がるかもしれない。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

例

数値的に降順(逆順)にソートする。

```
sort -nr
```

アルファベット順にソートし、第 1・第 2 フィールドは無視する。キーに開始フィールドとなる 3 だけを指定すれば、各行末までが比較される。

```
sort -k3
```

第 2 フィールドで数値的にソートし、同じになったものを第 5 フィールドの第 3 繰恒・4 文字で更にソートする。フィールドの区切りとして ':' を用いる。

```
sort -t : -k 2,2n -k 5.3,5.4
```

'-k 2,2' の代わりに '-k 2' と指定すると、sort は第 2 フィールドから行末までの全ての文字を、プライマリな「数値」キーとして扱ってしまう。通常の用途では、1 つ以上のフィールドにまたがったキーを数値的に扱うと、望む結果は得られないだろう。

'n' の指定が最初のキーの末尾で行われている点にも注意してほしい。これは '-k 2n,2' や '-k 2n,2n' としても効果は同じである。'b' 以外の全てのオプション指定は、開始フィールドに置いてもキー指定全体

の末尾に置いても、指定全体に効果を及ぼす。

パスワードファイルを第 5 フィールドでソートし、先頭の空白文字は無視する。第 5 フィールドが同じ値を持つ行は、第 3 フィールドのユーザーID で数値的にソートする。

```
sort -t : -k 5b,5 -k 3,3n /etc/passwd
```

数値比較オプション-n はグローバルに用いても結果は同じ。

```
sort -t : -n -k 5b,5 -k 3,3 /etc/passwd
```

英大文字小文字の違いを無視してソートされた tags ファイルを生成する。

```
find src -type f -print0 | sort -t / -z -f | xargs -0 etags –append
```

この‘-print0’、‘-z’、‘-0’は、改行(line feed)文字を含むパス名がソート操作によって壊れないようにするためのものである。

最後の例。フィールドの先頭・末尾の空白群を無視するには、第 1 キーの末尾フィールド指定もして、‘b’オプションを使えばよい。

```
sort -t : -n -k 5b,5b -k 3,3 /etc/passwd
```

あるいはグローバルな指定を-n の代わりに-b にして、第 2 キーのオプションに‘n’を追加するかたちでもよい。

```
sort -t : -b -k 5,5 -k 3,3n /etc/passwd
```

乗数

数値の後には、倍数を指定するサイズ指定文字と、通常のバイトを意味する B または 10 進の「商業用」バイトを意味する D を続けて置くことができる。たとえば‘1KB’は‘1024’と等しく、‘1KD’は‘1000’と等しい。この例外は b(512 バイト)、c(1 バイト)、w(使うべきでない-System V では 2 を意味し、4.2BSD では 4 を意味する)の 3 つで、これらの後に B や D をおくことはできない。

- k　キロ:通常バイト指定なら 2^10=1024、10 進バイト指定なら 10^3=1000
- M　メガ:2^20=1,048,576 または 10^6=1,000,000
- G　ギガ:2^30=1,073,741,824 または 10^9=1,000,000,000
- T　テラ:2^40=1,099,511,627,776 または 10^12=1,000,000,000,000
- P　ペタ:2^50=1,125,899,906,842,624 または 10^15=1,000,000,000,000,000
- E　エクサ:2^60=1,152,921,504,606,846,976 または 10^18=1,000,000,000,000,000,000
- Z　ゼタ:2^70=1,180,591,620,717,411,303,424 または 10^21=1,000,000,000,000,000,000,000
- Y　ヨタ:2^80=1,208,925,819,614,629,174,706,176 または 10^24=1,000,000,000,000,000,000,000,000

移植性

sort の歴史的な(BSD と SystemV の)実装では、いくつかのオプション(特に-b、-f、-n)の解釈が異なる。POSIX に従えば、-n はもはや-b を暗黙のうちに指定することはない。このあたりを首尾一貫させるため、-M も同様に変更されている。これによって、曖昧な指定ではフィールド内の文字位置の指定の意味が変わってしまうかもしれない。唯一の解決法は、-b を明示的に指定することである。

ロケール

- LC_COLLATE

  (特に他の指定がない限り)全ての比較で用いられる文字の照合順序を指定する。
- LC_CTYPE

  -b、-d、-f、-i といった出力順序オプションの動作に影響する。
- LC_NUMERIC

  基数文字と桁区切り文字(,など)を指定する。
- LC_TIME

  月名のスペルを決める。-M に影響する。

6.2.5.27 ファイルからコマンドを読み込み実行する（source）

書式

```
. filename [arguments]
source filename [arguments]
```

説明

filename からコマンドを読み込み、現在のシェル環境のもとで実行します。そして filename 中で最後に実行したコマンドの終了ステータスを返します。filename にスラッシュが含まれていない場合、filename は PATH に含まれるディレクトリから探されます。PATH 内で検索されるファイルは、実行可能である必要はありません。bash が posix モードで動作していれば、PATH 中でファイルを見つけられなかった場合には、カレントディレクトリが検索されます。組み込みコマンド shopt に対する sourcepath オプションが無効にされている場合、PATH の検索は行われません。何らかの arguments が与えられている場合、これらの引数は filename を実行した時の位置パラメータとなります。そうでない場合は、位置パラメータは変更されません。返り値はスクリプト内で最後に実行したコマンドのステータスです。(コマンドが全く実行されなければ 0 です)。また filename が見つからない場合や読み込めない場合には偽となります。

6.2.5.28 ファイルを決まった大きさに分割する (split)

書式

split [-LINES] [-a LENGTH] [-b BYTES] [-l LINES] [-C BYTES] [--bytes=BYTES] [--lines=LINES] [--line-bytes=BYTES] [--verbose] [FILE]

split [--help] [--version]

説明

split は FILE の各セクションの内容を持つファイルを次々に作成して出力する。FILE が与えられなかったり '-' だった場合には標準入力が用いられる。

デフォルトでは、split は FILE のうちの 1000 行ずつ(1000 行に満たない場合は残り全部)を各出力ファイルに書き込む。

出力ファイルの名前は PREFIX(デフォルトは 'x')に 'aa' や 'ab' などの文字列集合を付加したものになる。出力ファイルをファイル名でソートして結合すると元のファイルになるように、付加する文字列が選ばれる。(676 よりも多くの出力ファイルが必要になる場合は、split は 'zaa', 'zab' などを用いる。)このグループの長さは--suffix-length で変更できる。

オプション

- -LINES、-l LINES、--lines=LINES

  FILE の LINES 行分ずつ(デフォルトは 1000 行)を各出力ファイルに書き込む。短いオプション形式-LINES は推奨されない。
- -a LENGTH、--suffix-length=LENGTH

  LENGTH 文字のサフィックスを使う(デフォルトは 2 である)。
- -b BYTES、--bytes=BYTES

  FILE の BYTES バイトずつを各出力ファイルに書きこむ。'b' を付加すると BYTES は 512 倍、'k' なら 1024 倍、'm' なら 1048576 倍される。
- -C BYTES、--line-bytes=BYTES

  FILE の行を BYTES バイトを越えない範囲で、できるだけたくさん各出力ファイルに書き込む。BYTES バイトよりも長い行があった場合には、その行の残りが BYTES バイト未満になるまで BYTES バイト毎を出力ファイルに書き込み、後は通常に動作を続ける。BYTES は--bytes オプションと同様の形式で指定できる。
- -v、--verbose

  常にファイル名のヘッダを表示する。
- --help

標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.29 ファイル中の表示可能な文字列を表示する (strings)

書式

```
strings [-a|-|--all] [-f|--print-file-name] [-o] [--help] [-v|--version]
[-n min-len|-min-len|--bytes=min-len] [-t {o,x,d}[--target=bfdname] |--radix={o,x,d}]
file
```

説明

GNUstrings は、与えられたそれぞれの file から表示可能なキャラクタの列を表示する。デフォルトでは 4 文字以上の長さのものを、可能な限り表示するが、この長さは以下で説明するオプションによって変更可能である。ファイルがオブジェクトファイルの場合のデフォルトでは、初期化・ロードされたセクションからの文字だけを表示する。他のファイルについては、ファイル全体から文字列を探し、表示する。

strings は主としてテキストファイル以外のファイル内容を判断するために用いられる。

オプション

- -a、--all、-

  オブジェクトファイルから初期化・ロードされたセクションのみをスキャンするのではなく、ファイル全体をスキャンする。

- -f、--print-file-name

  それぞれの文字列の前にファイル名を表示する。

- --help

  strings のオプションの要約を標準出力に表示して終了する。

- -v、--version

  strings のバージョン番号を標準出力に表示して終了する。

- -n min-len、-min-len、-bytes=min-len

  min-len 以上の長さを持つ文字のシーケンスを表示する。デフォルトは 4。

- -t {o,x,d}、--radix={o,x,d}

  それぞれの文字列の前にファイル中のオフセットを表示する。引数として与えられる一文字で、オフセットの基数を(8、16、10)進数で与えるように指定する。

- --target=bfdname

  オブジェクトコードのフォーマットを、システムのデフォルト以外のものに指定する。指定できる

フォーマットをリストするための方法については objdump(1)を見よ。

- -o

  -t o と同じ。

関連項目

info の 'binutils' エントリ、The GNU Binary Utilities, Roland H. Pesch (October 1991)、 ar(1)、 nm(1)、 objdump(1)、 ranlib(1)

6.2.5.30 ファイルを結合して逆順に表示する (tac)

書式

tac [-br] [-s SEPARATOR] [--before] [--regex] [FILE...]

tac [--help] [--version]

説明

tac は指定されたファイルの内容を標準出力に書き込む。その際、レコードを逆順にする(レコードのデフォルトは行単位)。FILE が一つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

レコードはセパレータ文字列によって分離される(デフォルトは改行文字)。デフォルトでは、セパレータ文字列は(入力ファイル中で後置されていた)レコードの最後に付加されて出力される。

DOS プラットフォームでは、tac はファイルをバイナリモードで読み込む。詳細は cat(1)を見よ。

オプション

- -b、--before

  セパレータ文字列を(入力ファイル中で前置されていた)レコードの先頭に付加する。

- -r、--regex

  セパレータ文字列を正規表現として取り扱う。

- -s SEPARATOR、--separator=SEPARATOR

  改行文字の代わりに、STRING をレコードのセパレータに用いる

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.31 ファイルの末尾の部分を表示する (tail)

書式

```
tail [<-|+>COUNTOPTIONS] [-fqvF] [-c BYTES] [-n LINES] [--follow][--bytes=BYTES]
[--lines=LINES] [--follow[=HOW]] [--max-consecutive-size-changes=NUMBER]
[--max-unchanged-stats=NUMBER] [--pid=PID] [--retry] [--sleep-interval=SECS] [--quiet]
[--silent]
[--verbose] [FILE...]

tail [--help] [--version]
```

説明

tail は引数に与えられた FILE それぞれの末尾の部分(デフォルトでは 10 行)を表示する。FILE が 1 つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

2 つ以上の FILE が指定されると、tail は以下のような 1 行のヘッダを各出力 FILE の前に表示する:

==> FILENAME <==

tail は 2 つのオプション形式を受付ける。新しい形式は、数値をオプションの引数として与える('-n 1')ものであり、古い形式はあらゆる文字オプションの前に数値を指定する('-1' や '+1')ものである。

オプション引数に数値 N が '+' を前置して指定された場合は、tail は各ファイルの先頭から N 番目の項目以降を表示する。通常の動作では末尾から数える。この構文は推奨されない。

FILE が切り詰められた場合は、tail はファイルが短くなったことを検知し、新たなファイル末尾に移動し、そこから読み込みを始める。

tail では出力の大きさの指定に制限が無い(GNU 版以外の tail にはそうでないものもある)。また-r オプション(逆順表示)は無い。ファイルを逆順に表示するのは、ファイルの末尾を表示するのとは本質的に異なるからである。BSD 版 tail(-r オプションを持っている)はバッファ(通常 32k)よりも小さいファイルを逆順表示できるに過ぎない。この目的には GNU 版の tac コマンドを用いる方が、信頼性も融通性も高い。

オプション

- <-|+>COUNTOPTIONS

  このオプションは最初に指定した場合に限って認識される。COUNT は 10 進数の数値。単位を表す文字('b','k','m')を後置したり(それぞれの意味は '-c' のものと同じ)、行単位のカウントを指定する 'l' を後置したり、他のオプション文字('cqv')を後置したりできる。何も文字が後置されなかった場合は 'l' が指定されたのと同じことになる。短いオプション形式-と+は推奨さ

れない。

- -c BYTES、--bytes=BYTES

  行単位ではなく、末尾の BYTES バイトを表示する。'b' を追加すると BYTES の 512 倍、'k' は 1024 倍、'm' は 1048576 倍を指定したことになる。

- -f、--follow[=HOW]

  ファイルの内容がどんどん増え続けているものと仮定し、ファイルの最終部分の文字を読み続けようと無限にループする。パイプから読み込んでいる場合は無視される。2 つ以上のファイルが指定された場合は、tail は異なったファイルの追加分を受け付けるごとにヘッダを表示し、出力がどのファイルに由来するものかがわかるようにする。ファイルの追跡方法を指定することもできる。

  descriptor　ファイル末尾の追跡を、ファイルが unlink されたり rename されたりした後に行う。つまり永遠に行う。

  name　ファイル末尾の追跡を、ファイルが remove された後に行う。tail は FILE が(unlink や rename によって)もう存在しないことを確認すると、もう一度オープンしようと試みる。

- -n LINES、--lines=LINES

  末尾の LINES 行を表示する。

- -q、--quiet、--silent

  ファイル名のヘッダを出力しない。

- -v、--verbose

  常にファイル名のヘッダを出力する。

- -F

  --follow=name --retry と等価である。

- --max-consecutive-size-changes=NUMBER

  名前で追跡するとき、連続 NUMBER 回のサイズ変更が検知されるまで (rename や remove されても)追跡する。検知されたらファイル名がまだ以前と同じ「デバイス/i-ノード」の組み合わせに対応しているかどうかを、ファイルをオープンして fstat することによって調べる。デフォルトは 200。

- --max-unchanged-stats=NUMBER

  名前で追跡するとき、連続 NUMBER 回サイズが変わらなかったら、ファイル名がまだ以前と同じ「デバイス/i-ノード」の組み合わせに対応しているかどうかを、ファイルをオープンして fstat することによって調べる。デフォルトは 5。

- --pid=PID

  ファイルを追跡するとき、tail は pid が PID のプロセスが終了したら終了する。そのようなプロセスがなければ、1 回分のループを行って終了する。

- --retry

  tail がファイルを名前で追跡していて、ファイルがなくなったことを検知したら、再オープンを成

功するまで繰り返す。このオプションを指定しなければ、tail は単にファイルが存在しないことを報告して以降のチェックを行わない。

- --sleep-interval=SECS

  追跡しているファイルのチェックを SECS 秒ごとに行う(デフォルトは 1 秒)。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.32 ログファイルの追加分を追跡する (tailf)

書式

tailf file

説明

tailf はファイルの末尾 10 行を表示し、その後ファイルに追加される行を待ち続ける。 tailf は tail -f と似ているが、ファイルへの追加書き込みがないとファイルへアクセスを行わない。したがってファイルのアクセス時間は更新されないので、ログに関る動作が起らない間は、定期的なファイルシステムのフラッシュも生じない。

tailf は、ラップトップで、あまり頻繁に書き込まれない状態のログファイルを監視にする場合に大変便利である。ハードディスクをスピンダウンさせ、電池を長持ちさせることができる。

6.2.5.33 有向グラフのトポロジカルなソートを行う (tsort)

書式

tsort

tsort [--help] [--version]

説明

tsort は与えられた FILE の内容をトポロジカルにソートする。FILE が与えられないと標準入力から読み込む。また FILE が '-' だった場合には標準入力が用いられる。

tsort 空白で区切られた文字列の組み合わせを FILE から読み込み、各組み合わせの順序から全体の順序関係を求めて標準出力に書く。循環が見付かった場合には、最初の循環を標準出力に書く。

オプション

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

例

tsort に"X は Y より前に起こった"というペアを与え、それがソートできるかを見てみよう。各行に与える部分的な順序は、必ずしも全体の順序で隣接しているわけではないことに注意。

```
tsort << EOF
neolitic bronze
greeks linux
pyramids greeks
bronze pyramids
bronze greeks
EOF
```

出力は以下となる。

```
neolitic
bronze
pyramids
greeks
linux
```

6.2.5.34 スペースをタブに変換する (unexpand)

書式

```
unexpand [-a] [-TAB1[,TAB2...] [-t TAB1[,TAB2...] [--all]
[--first-only] [--tabs=TAB1[,TAB2...] [FILE...]

unexpand [--help] [--version]
```

説明

unexpand は指定されたファイルの内容を標準出力に書き込む。その際「2 つ以上のスペースまたはタブ文字からなる文字列」を変換し、できるだけ多くのタブ文字に、必要な分のスペース文字を加えた文字列にする。FILE が 1 つも与えられないと標準入力から読み込む。また FILE が '-' だった場合には、そのファイルには標準入力が用いられる。

オプション

- -TAB1[,TAB2...]、-t TAB1[,TAB2...]、--tabs=TAB1[,TAB2...]

  タブストップが 1 つだけ与えられた場合には、タブを TAB1 おきに設定する(デフォルトは 8)。その他の場合は、タブを TAB1 桁、TAB2 桁...に設定し(桁数は 0 から数え始める)、最後のタブ位置以降のタブ文字やスペース文字は変換せずにそのまま出力する。タブストップを--tabs(-t)オプションで指定する際には、それぞれの間は空白またはコンマで区切る。短いオプション形式-は推奨されない。このオプションが指定されると-a オプションも同時に指定されたことになる。

- -a、--all

  先頭のものだけでなく、行中の「タブ文字とスペース文字からなる 2 文字以上の文字列」をすべて変換する。

- --first-only

  1 度だけ置き換えを行って終了する。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.2.5.35 ソートされたファイルから重なった行を削除する

書式

```
uniq [<+|->N] [-ciduD] [-f N] [-s N] [-w N] [--all-repeated[={prepend,separate,none}]]
[--check-chars=N]    [--count]    [--ignore-case]    [--repeated]    [--skip-chars=N]
[--skip-fields=N]
[--unique] [INFILE [OUTFILE]]

uniq [--help] [--version]
```

説明

uniq は指定された INFILE にあるユニークな(＝他と内容の重ならない)行を標準出力(OUTFILE が指定されていたらそれ)に書き出す。INFILE が与えられなかったり '-' だった場合には、標準入力が用いられる。

デフォルトでは、uniq はソートされたファイルにあるユニークな行を表示する。つまり複数の行が同一な内容を持つ場合は、1 行だけ表示して残りは捨てる。オプションで指定すると、1 回しか現われない行だけを表示したり、複数回現われる行だけを表示することもできる。

uniq に与える入力はソートされていなければならない。入力がソートされていない場合は、'sort-u' を使うのが良いだろう。

オプション

- -N、-f N、--skip-fields=N

  同一行かどうかの判断を行う前に N 個のフィールドをスキップする。フィールドとは空白とタブ以外の文字からなる文字列で、フィールド間は 1 つ以上の空白かタブで区切られる。短いオプション形式-は推奨されない。

- +N、-s N、--skip-chars=N

  同一行かどうかの判断を行う前に N 個の文字をスキップする。フィールドスキップと文字スキップのオプションを両方指定した場合は、フィールドスキップが先に行われる。短いオプション形式+は推奨されない。

- -c、--count

  それぞれの行が何回現われたかを行の内容とともに表示する。

- -i、--ignore-case

  比較の際に英大文字小文字の違いを無視する。

- -d、--repeated

  同じ内容が 2 行以上あるものだけを出力する。

- -u、--unique

  1 回しか現われない行だけを出力する。

- -w N、--check-chars=N

  行を比較するとき、各行の N 個の文字だけを使う。これはフィールドや文字のスキップを行った後の数である。デフォルトでは、スキップ後残った文字すべてが比較の対象にされる。

- -D、--all-repeated[=[prepend,separate,none]]

  重複行をすべて表示し、重複しなかった行は表示しない。prepend が指定された場合、重複した行のグループの前に改行を出力する。separate は、最初のグループの前に改行を出力しない以外は、prepend と同じである。このオプションは、主に他のオプションと組み合わせて使う。例えば大文字小文字を無視して比較したり、特定のフィールドだけで比較するような場合である。このオプションは GNU による拡張である。

- --help

  標準出力に使用方法のメッセージを出力して正常終了する。

- --version

  標準出力にバージョン情報を出力して正常終了する。

6.3. システムコール

6.3.1. アクセス制御属性管理

6.3.1.1 ユーザー識別(identity)を設定する (setuid)

書式

```
#include <sys/types.h>
#include <unistd.h>

int setuid(uid_t uid);
```

説明

setuid は現在のプロセスの実効(effective)ユーザーID を設定する。もし呼び出したプロセスの実効ユーザーID が root ならば、実(real)ユーザーID と保存(saved)ユーザーID も設定される。

Linux では、setuid は_POSIX_SAVED_IDS をもった POSIX 版のように実装されている。これは(ルート以外の)setuid プログラムにそのユーザーの特権を全て与え、特権の必要ない仕事をし、本来の実効ユーザーID に安全な方法で再び戻すことを許す。

ユーザーがルート(root)またはプログラムがルートに setuid されているならば、特別の注意が払われる。setuid 関数は呼び出し者の実効 uid をチェックし、それがスーパー・ユーザーならば、プロセスに関連する全てのユーザーID に uid を設定する。これが行なわれた後にはプログラムが再びルートの特権を得ることはできない。

このように、setuid-root プログラムは一時的にルート特権を与え、ルートでないように振舞うことができ、それから setuid を使って再びルート特権を得ることができないようにする。(POSIX でない、BSD)コールは seteuid で行なうことができる。

返り値

成功した場合はゼロが返される。エラーの場合は-1 が返され、errno が適切に設定される。

エラー

- EPERM

  ユーザーがスーパー・ユーザー(super-user)でなく、uid が呼び出したプロセスの実ユーザーID

または保存ユーザーIDと一致しない。

- EAGAIN

  uidが現在のユーザーIDとマッチせず、このuidによってプロセスがNPROC rlimitを超えた。

準拠

SVr4、SVID、POSIX.1。4.4BSD のコールとは完全な互換性はない、BSD のコールは実(real)、保存(saved)、実効(effective)IDの全てを設定する。SVr4には他にEINVALエラーについての記述がある。

LINUX 特有の注意

Linuxはファイル・システム・ユーザーIDの概念を持つ。通常、これは実効ユーザーIDに等しい。setuidコールは現在のプロセスのファイル・システム・ユーザーIDも設定する。setfsuid(2)も参照すること。

uidが昔の実効uidと異っていた場合、プロセスはコア・ダンプすることを禁止される。

関連項目

getuid(2)、setreuid(2)、seteuid(2)、setfsuid(2)

6.3.1.2 グループ識別(identity)を設定する (setgid)

書式

```
#include <sys/types.h>
#include <unistd.h>

int setgid(gid_t gid);
```

説明

setgidは現在のプロセスの実効(effective)グループIDを設定する。もしスーパー・ユーザーによって呼び出された場合は、実(real)グループIDと保存(saved)グループIDも設定される。

Linuxにおいて、setgidは_POSIX_SAVED_IDSをもった POSIX 版のように実装されている。これは suid root でない setgid プログラムにそのグループの特権の全て落とし、特権の必要ない仕事をし、本来の実効グループIDに安全な方法で再び戻すことを許す。

返り値

成功した場合はゼロが返される。エラーの場合は-1 が返され、errno が適切に設定される。

エラー

- EPERM

  ユーザーがスーパー・ユーザーでなく(CAP_SETGID 資格がなく)、gid が呼び出したプロセスの実効グループ ID または保存セットグループ ID と一致しない。

準拠

SVr4、SVID.

関連項目

getgid(2)、setregid(2)、setegid(2)

6.3.1.3 実効ユーザーID と実効グループ ID を設定する (seteuid、setegid)

書式

```
#include <sys/types.h>
#include <unistd.h>

int seteuid(uid_t euid);
int setegid(gid_t egid);
```

説明

seteuid はカレントプロセスの実効ユーザーID を設定する。非特権ユーザーのプロセスは、ユーザーID を実ユーザーID・実効ユーザーID・保存ユーザーID にしか設定できない。

setegid でも、「ユーザー」の代わりに「グループ」に対して全く同じことが行われる。

返り値

成功した場合は 0 が返される。エラーの場合は-1 が返され、errno が適切に設定される。

エラー

- EPERM

カレントプロセスがスーパーユーザーのものではなく、euid(または egid)が実ユーザー(グループ)ID、または実効ユーザー(グループ) ID、または保存ユーザー(グループ)ID ではない。

注意

実効ユーザー(グループ)ID を保存ユーザー(グループ)ID に設定することは、Linux 1.1.37(1.1.38)から可能になった。全てのシステム上で、_POSIX_SAVED_IDS をチェックすべきである。

libc4、libc5、glibc2.0 では seteuid(euid)は setreuid(-1、euid)と同じなので、保存ユーザーID を変更する。glibc2.1 ではsetresuid(-1, euid,-1)と同じなので、保存ユーザーID を変更しない。同様な注意点が setegid にもある。

準拠

BSD 4.3

関連項目

geteuid(2)、setuid(2)、setreuid(2)、setresuid(2)

6.3.1.4 ファイルシステムのチェックに用いられるユーザ ID を設定する (setfsuid)

書式

```
#include <unistd.h> /* glibc では <sys/fsuid.h> */
int setfsuid(uid_t fsuid);
```

説明

setfsuid は Linux カーネルがファイル・システムに対する全てのアクセスのチェックに使用するユーザ ID を設定する。通常は fsuid の値は実効(effective)ユーザ ID と同じになる。実際、実効ユーザ ID が変更される度に fsuid もまた新しい実効ユーザ ID の値に変更される。

通常、setfsuid や setfsgid が明示的に呼び出されるのは、LinuxNFS サーバーのように、ファイル・アクセスに用いるユーザ ID/グループ ID を変更しなければならないが、対応する実(real)/実効(effective)ユーザ ID/グループ ID は変更したくないようなプログラムに限られる。NFS サーバーのようなプログラムで、通常のユーザ ID を変更すると、プロセスを望まないシグナルにさらす可能性があり、セキュリティ・ホールになる。(下記参照)

setfsuid はスーパー・ユーザによって呼び出された場合や、fsuid が実ユーザ ID、実効ユーザ ID、保存セットユーザ ID(saved set-user-ID)、現在の fsuid の値のいずれかに一致する場合にのみ成功する。

返り値

成功した場合、fsuid の以前の値を返す。エラーの場合は fsuid の現在の値を返す。

準拠

setfsuid は Linux 特有であり、移植を想定したプログラムで使用してはいけない。1.1.44 以降の Linux カーネルと 4.7.6 以降の libc に存在する。

バグ

いかなる種類のエラー・メッセージも呼び出し元に返さない。呼び出しが失敗した時は、最低でも EPERM くらいは返すべきである。

注意

glibc が引き数が uid として不正だと判断した場合はシステム・コールを行わず errno に EINVAL を設定して-1 が返される。

このシステムコールが導入された当時、プロセスは同じ実効ユーザ ID のプロセスへシグナルを送ることができた。今日では、シグナル送信権限の扱いはかなり違うものになっている。

関連項目

kill(2)、setfsgid(2)

6.3.1.5 実(real)と実効(effective)ユーザー(グループ)ID を設定する (setreuid、setregid)

書式

```
#include <sys/types.h>
#include <unistd.h>
int setreuid(uid_t ruid, uid_t euid);
int setregid(gid_t rgid, gid_t egid);
```

説明

setreuid はカレントプロセスの実(real)ユーザーID と実効(effective) ユーザーID を設定する。非特権ユーザーは、実ユーザーID を実ユーザーID または実効ユーザーID にしか設定できず、実効ユーザーID を実ユーザーID または実効ユーザーID または保存(saved)ユーザーID にしか設定できない。

実ユーザーID や実効ユーザーID に-1 を与えた場合、システムはその ID を変更しない。

実ユーザーID が設定されたり、実効ユーザーID が前の実ユーザーID と異った値に設定された場合、保存ユーザーID には新しい実効ユーザーID の値が設定される。

これと全く同様に、setregid はカレントプロセスの実グループ ID と実効グループ ID を設定し、上記の説明で「ユーザー」を「グループ」に読み替えたことが成り立つ。

返り値

成功した場合はゼロが返される。エラーの場合は-1 が返され、errno が適切に設定される。

エラー

- EPERM

  現在のプロセスがスーパー・ユーザーでなく、変更が(i)実効ユーザー(グループ)ID と実ユーザー(グループ)ID を入れ換える。(ii)片方の値を他方に設定する。(iii)実効ユーザー(グループ)ID に保存ユーザー(グループ)ID の値を設定する。のいずれでもない。

注意

実効ユーザー(グループ)ID を保存ユーザー(グループ)ID に設定することは、Linux1.1.37(1.1.38)から可能になった。

準拠

BSD4.3(setreuid と setregid 関数コールは 4.2BSD で登場した)。

関連項目

getuid(2)、getgid(2)、setuid(2)、setgid(2)、seteuid(2)、setresuid(2)

6.3.1.6 ファイルのモードを変更する (chmod、fchmod)

書式

```
#include <sys/types.h>
#include <sys/stat.h>

int chmod(const char *path, mode_t mode);
int fchmod(int fildes, mode_t mode);
```

説明

path で与えられたファイル、または fildes で参照されるファイルのモードを変更する。モードは、次の各モードの or をとったもので指定する:

- S_ISGID - 02000 実行時のセット・グループ・ID (set group ID)
- S_ISVTX - 01000 スティッキー (sticky) ビット
- S_IRUSR (S_IREAD) - 00400 所有者 (owner) による読み取り (read)
- S_IWUSR (S_IWRITE) - 00200 所有者による書き込み (write)
- S_IXUSR (S_IEXEC) - 00100 所有者による実行 (execute) / 検索 (search)
- S_IRGRP - 00040 グループによる読み取り
- S_IWGRP - 00020 グループによる書き込み
- S_IXGRP - 00010 グループによる実行/検索

プロセスの実効(effective)UID がゼロであるか、ファイルの所有者と一致しなければならない。

もしプロセスの実効 UID がゼロでなく、ファイルのグループ ID がプロセスの実効グループ ID または補助的なグループ ID にマッチしない場合、S_ISGID ビットはオフにされるが、これによってエラーが返されることはない。

ファイル・システムによっては、ファイルの書き込みを行う時にセット・ユーザーID とセット・グループ ID ビットと実行ビットがオフにされることがある。ファイル・システムの中には、スーパー・ユーザーだけが特別の意味を持つスティッキー・ビットを設定できるものがある。スティッキー・ビットとディレクトリに対するセット・ユーザー(グループ)・ID ビットについては、stat(2)を見よ。

NFS ファイルシステム上では、パーミッションを制限すると、既にオープンされているファイルに対してすぐに影響が及ぶ。これはアクセス制御がサーバー上で行われているが、オープンされているファイルはクライアント側で管理されているためである。クライアント側でファイル属性のキャッシュが有効になっている場合に、パーミッションの制限を緩くすると、他のクライアントに情報が伝わるのが遅れるかもしれない。

返り値

成功すると、0 を返す。失敗すると、-1 を返し、errno に適切な値を設定する。

エラー

ファイル・システムによっては他のエラーを返す場合がある。chmod で一般的なエラーを以下に挙げる。

- EPERM
  実効 UID がファイルの所有者と一致せず、スーパー・ユーザーでもない。
- EROFS

ファイルが読み込み専用(read only)のファイル・システム上にある。

- EFAULT

  path が割り当てられたアドレス空間外を指している。

- ENAMETOOLONG

  path が長過ぎる。

- ENOENT

  ファイルが存在しない。

- ENOMEM

  カーネルに十分なメモリがない。

- ENOTDIR

  パス名の構成要素がディレクトリではない。

- EACCES

  パス名の構成要素に検索許可がない。

- ELOOP

  path を解決する際に遭遇したシンボリック・リンクが多過ぎる。

- EIO

  I/O エラーが発生した。

fchmod で一般的なエラーを挙げる:

- EBADF

  ファイル・ディスクリプターfildes が有効でない。

- EROFS

  上記を参照。

- EPERM

  上記を参照。

- EIO

  上記を参照。

準拠

chmod は SVr4、SVID、POSIX、X/OPEN、4.4BSD と互換性がある。SVr4 は EINTR、ENOLINK、EMULTIHOP は返すが ENOMEM は返さない。POSIX.1 には EFAULT、ENOMEM、ELOOP、EIO エラーや S_IREAD、S_IWRITE,S_IEXEC マクロについての記述はない。

fchmod は 4.4BSD、SVr4 と互換性がある。SVr4 には EINTR、ENOLINK エラー状態の記述はない。POSIX では fchmod 関数は少なくとも_POSIX_MAPPED_FILES と_POSIX_SHARED_MEMORY_OBJECTS のどちらかが定義されている必要がある。そして ENOSYS と EINVAL エラー状態についての記述はあるが、EIO

についての記述はない。

POSIX と X/OPEN にはスティッキー・ビットについての記述はない。

関連項目

open(2)、chown(2)、execve(2)、stat(2)

6.3.1.7 ファイルの所有者を変更する (chown、fchown、lchown)

書式

```
#include <sys/types.h>
#include <unistd.h>

int chown(const char *path, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
int lchown(const char *path, uid_t owner, gid_t group);
```

説明

path または、fd で指定されたファイルの所有者(owner)を変更する。スーパー・ユーザーだけがファイルの所有者を変更できる。ファイルの所有者は、その所有者が属しているグループのいずれかにファイルのグループを変更することができる。スーパー・ユーザーは、任意のグループに変更できる。もし、owner または group に-1 が指定された場合、それらの ID は変更されない。

スーパー・ユーザー以外によって実行ファイルの所有者またはグループが変更された場合は S_ISUID と ISGID モードビットはクリアされる。POSIX はこの動作やルートが chown を行なった場合については特に指定されていない。Linux における動作はカーネルのバージョンに依存する。非グループ実行ファイル(S_IXGRP ビットが設定されていない)の場合には S_ISGID ビットは強制ロック(mandatory locking)を意味している。そしてそれは chown ではクリアできない。

返り値

成功すると、0 を返す。失敗すると、-1 を返し、errno に適切な値を設定する。

エラー

ファイルシステムによっては他のエラーが返される事がある。chmod で一般的なエラーを以下に挙げる:

- EPERM

  実効(effective)UID がファイルの所有者と一致せず、0 でもない。または、owner または group

が正しく指定されていない。

- EROFS

  指定したファイルが読み込み専用(read-only)のファイル・システム上にある。

- EFAULT

  path が割り当てられたアドレス空間外を指している。

- ENAMETOOLONG

  path が長過ぎる。

- ENOENT

  ファイルが存在しない。

- ENOMEM

  カーネルに十分なメモリがない。

- ENOTDIR

  path の構成要素がディレクトリでない。

- EACCES

  path の構成要素に検索許可(search permission)がない。

- ELOOP

  path を解決する際に遭遇したシンボリック・リンクが多過ぎる。

fchown で一般的なエラーを以下に挙げる:

- EBADF

  ディスクリプターが有効でない。

- ENOENT

  上記を参照。

- EPERM

  上記を参照。

- EROFS

  上記を参照。

- EIO

  i ノード(inode)を修正する際に低レベル I/O エラーが発生した。

注意

Linux の 2.1.81 より前のバージョン(特に 2.1.46 以前)では、chown はシンボリック・リンクを追跡しない。Linux2.1.81 以降では chown はシンボリック・リンクを追跡し、新たなシステム・コール lchown はシンボリック・リンクを追跡しない。Linux 2.1.86 以降ではこの新しいコール(古い chown と全く同じ動作を行なう)は同じシステムコール番号を持ち chown は新しく導入された番号を持つ。

fchown は_BSD_SOURCE が定義されている場合のみ利用できる。(明示的にもしくは暗黙に_POSIX_SOURCE を定義せず、-ansi フラグを指定してコンパイルしなければ_BSD_SOURCE が自動的に定義される)

準拠

chown は SVr4、SVID、POSIX、X/OPEN と互換性がある。4.4BSD 版ではスーパー・ユーザーのみが使用できる。SVr4 には EINVAL、EINTR、ENOLINK, EMULTIHOP を返すと記述されているが、ENOMEM はない。POSIX.1 には ENOMEM、ELOOP について記述はない。

fchown は 4.4BSD、SVr4 と互換性がある。SVr4 には他に EINVAL、EIO、EINTR、ENOLINK エラー状態についての記述がある。

制限

chown()方式は UID マッピングを使用した NFS ファイル・システムを侵害する。さらにファイルの内容にアクセスする全てのシステム・コールを侵害する。これは chown()が既にオープンされたファイルに対するアクセスをただちに取り消すことによる。クライアント側のキャッシュにより所有権が変更されてユーザーのアクセスが許した時点と、実際に他のクライアントでユーザーによってファイルにアクセスできる時点との間に時間差があるかもしれない。

関連項目

chmod(2)、flock(2)

6.3.1.8 プロセス・グループ(process group) の設定/取得を行なう (setpgid,、getpgid,、setpgrp、getpgrp)

書式

```
#include <unistd.h>

int setpgid(pid_t pid, pid_t pgid);
pid_t getpgid(pid_t pid);
int setpgrp(void);
pid_t getpgrp(void);
```

説明

setpgid は pid で指定したプロセスのプロセス・グループ ID に pgid を設定する。もし pid がゼロならば、呼び出したプロセスのプロセス ID が pid として使用される。もし pgid がゼロならば、pid で指定されたプロセ

スのプロセス ID が pgid として使用される。setpgid をプロセスをあるプロセス・グループから別のグループへ移動するために使用する場合は(一部のシェルはパイプラインを生成する時にこれを行う)、両方のプロセス・グループは同じセッションの一部でなければならない。この場合は pgid は参加すべき既存のプロセス・グループを指定し、そのセッション ID は参加するプロセスのセッション ID に一致しなければならない。

getpgid は pid で指定されたプロセスのプロセス・グループ ID を返す。もし pid がゼロならば、呼び出したプロセスのプロセス ID が pid として使用される。

setpgrp()呼び出しは setpgid(0,0)と等価である。

同様に、getpgrp()は getpgid(0)に等しい。各プロセス・グループはセッションのメンバーであり、各プロセスはそのプロセス・グループがメンバーであるセッションのメンバーである。

プロセス・グループはシグナルを配送する時や、端末(terminal)によって入力の要求を調停(arbitrate)する時に使用される:端末と同じプロセス・グループ ID を持つプロセスはフォアグランドとして読み込みを行なうことができ、その他のプロセスが端末から読み込みを行なおうとした場合、シグナルによって中断(block)させられる。

これらのコールは csh(1)のようなプログラムによって、ジョブ・コントロール(job control)の実装の際にプロセス・グループを作成するために使用される。termios(3)に記述されている TIOCGPGRP と TIOCSPGRP コールは制御端末のプロセス・グループを取得/設定するのに使用する。

もしセッションが制御端末を持っていて、CLOCAL が設定されておらず、ハングアップが起きた場合、セッション・リーダーに SIGHUP が送られる。セッション・リーダーが終了した場合にはその制御端末のフォアグランドのプロセス・グループに所属する全てのプロセスに SIGHUP シグナルが送られる。

プロセスの終了によってプロセス・グループが孤児になった場合で、もし新たに孤児になったプロセス・グループのメンバーが停止すると、新たに孤児になったプロセス・グループの全てのプロセスに SIGHUP シグナルに続けて SIGCONT シグナルが送られる。

返り値

setpgid と setpgrp は成功した場合、ゼロを返す。エラーの場合は-1 を返し、errno が適切に設定される。

getpgid は成功した場合プロセス・グループを返す。エラーの場合は-1 を返し、errno が適切に設定される。

getpgrp は常に現在のプロセスのプロセス・グループを返す。

エラー

- EINVAL

  pgid が 0 より小さい。(setpgid、setpgrp)

- EACCES

  呼び出し元プロセスの子プロセスのプロセスグループ ID を変更しようとしたが、すでにその子プロセスは execve を実行していた。(setpgid、setpgrp)

- EPERM

  プロセスを異なるセッションのプロセスグループに移動させようとした。または呼び出し元プロセスの子プロセスのプロセスグループ ID を変更しようとしたが、その子プロセスは別のセッションだった。またはセッションリーダーのプロセスグループ ID を変更しようとした。(setpgid、setpgrp)

- ESRCH

  getpgid の場合:pid がどのプロセスにも一致しない。setpgid の場合:pid がカレントプロセスではなく、カレントプロセスの子プロセスでもない。

準拠

setpgid 関数と getpgrp 関数は POSIX.1 に準拠している。setpgrp 関数は BSD4.2 に由来する。getpgid 関数は SVr4 に準拠している。

注意

POSIX の setpgid は BSD の setpgrp 関数からきている。また SysV にも同じ名前の関数があるが、これは setsid(2)と同じものである。

glibc でプロトタイプを得るためには、_XOPEN_SOURCE と_XOPEN_SOURCE_EXTENDED を両方とも定義するか、"#define_XOPEN_SOURCE n"を用いる。ここで n は 500 以上の整数である。

関連項目

getuid(2)、setsid(2)、tcgetpgrp(3)、tcsetpgrp(3)、termios(3)

6.4. ライブラリコール

6.4.1. パスワード管理

6.4.1.1 暗号化されたパスワードファイル用ルーチン（shadow）

書式

```
#include <shadow.h>
struct spwd *getspent();
struct spwd *getspnam(char *name);
void setspent();
void endspent();
struct spwd *fgetspent(FILE *fp);
struct spwd *sgetspent(char *cp);
int putspent(struct spwd *p, FILE *fp);
int lckpwdf();
int ulckpwdf();
```

説明

shadow は shadow パスワードファイル/etc/shadow の内容を操作するルーチンである。#include ファイルに与えられている構造体は以下の通り。

```
struct spwd {
            char *sp_namp; /* user login name */
            char *sp_pwdp; /* encrypted password */
            long sp_lstchg; /* last password change */
            int  sp_min; /* days until change allowed. */
            int  sp_max; /* days before change required */
            int  sp_warn; /* days warning for expiration */
            int  sp_inact; /* days before account inactive */
            int  sp_expire; /* date when account expires */
            int  sp_flag; /* reserved for future use */
       }
```

各フィールドの意味は:

- sp_namp - ヌル終端されたユーザ名文字列へのポインタ
- sp_pwdp - ヌル終端されたパスワード文字列へのポインタ
- sp_lstchg - 1970 年 1 月 1 日からパスワード最終変更日時迄の日数
- sp_min - パスワード変更が出来るようになるまでの日数
- sp_max - パスワードを変更しなくても良い日数
- sp_warn - パスワードが期限切れになる前に、期限切れが近づいている旨の警告をユーザに出

す期間の日数

- sp_inact – パスワードが期限切れになってから、アカウントが不能となり使用できなくなるまでの日数
- sp_expire – 1970 年 1 月 1 からアカウントが使用不能となる日迄の日数
- sp_flag – 将来使うときに向けて予約

getspent、getspname、fgetspent、sgetspent は、それぞれ struct spwd へのポインタを返す。getspent はファイルから次のエントリを、fgetspent は指定されたストリーム(正しい書式のファイルとみなされる)から次のエントリを返す。sgetspent は入力として与えられた文字列を用いて structspwd へのポインタを返す。getspnam はファイル中の現在の位置から name にマッチするエントリを探す。

setspent は shadow パスワードファイルへのアクセスを開始するために、endspent は終了するために用いられる。

/etc/shadow ファイルに対する排他的なアクセスを保証したい場合には、lckpwdf ルーチンと ulckpwdf ルーチンを用いる。lckpwdf は pw_lock を用いて最大 15 秒間ロックを取得しようとする。そして最初の 15 秒の残りの間、spw_lock によって二度目のロックをしようと試み続ける。計 15 秒間の間にいずれかの試みが失敗した場合は、lckpwdf は-1 を返す。いずれのロックも成功した場合は 0 が返される。

返り値

これらのルーチンは、エントリが残っていない場合や、処理の過程でエラーが発生した場合には NULL を返す。返り値が int であるルーチンは、成功したら 0 を、失敗したら-1 を返す。

警告

shadow されたパスワードファイルへのアクセスは制限されているので、これらのルーチンはスーパーユーザだけが利用できる。

ファイル

- /etc/shadow – 暗号化されたユーザパスワード

関連項目

getpwent(3)、shadow(5)

6.4.2. アクセス制御属性管理

6.4.2.1 ファイル作成マスクを設定する (umask)

書式

```
#include <sys/types.h>
#include <sys/stat.h>

mode_t umask(mode_t mask);
```

説明

umask システムコールは umask 値を mask & 0777 に設定する。

umask 値は open(2)で新しくファイルを作成する時に、ファイルの最初の許可(permission)を設定するために使用される。具体的には umask 値に設定されている許可が open(2)の mode 引き数から取り消される。(例えば、デフォルトの umask 値としては 022 という値がよく使用されるが、この時 mode に一般的な 0666 が与えられると、新たに作成されたファイルは 0666&~022=0644=rw-r--r--という許可を持つことになる。)

返り値

このシステムコールは必ず成功し、以前の umask 値を返す。

準拠

SVr4、SVID、POSIX、X/OPEN、BSD 4.3

関連項目

creat(2)、open(2)

6.4.3. 監査

6.4.3.1 システムロガーにメッセージを送る (closelog、openlog、syslog)

書式

```
#include <syslog.h>

void openlog(const char *ident, int option, int facility);
void syslog(int priority, const char *format, ...);
void closelog(void);
```

```
#include <stdarg.h>

void vsyslog(int priority, const char *format, va_list ap);
```

説明

closelog()はシステムのログ記録用プログラム(システムロガーsyslogd(8))への接続を終了する。closelog()は必須ではない。

openlog()はログを出力しようとしているプログラムからログ記録用プログラムへの接続を開始する。identで指定した文字列(通常は openlog()したプログラムの名前)はログ出力の一文一文に追加され、どのプログラムが出力したログかを識別するために使われる。option 引き数は、openlog()の動作とその後のsyslog()の呼び出しを制御するフラグを指定する。facility 引き数は、後で syslog()を呼び出す際に facilityが指定されなかった場合に使用されるデフォルト値を決定する。option と facility については後述する。openlog()は必須ではなく、必要に応じて syslog()から呼び出される。syslog()が呼び出した場合、ident のデフォルト値は NULL になる。

syslog()はログメッセージを出力し、syslogd(8)がそのメッセージを記録する。priority 引き数は facility とlevel との組合せで指定する。facility と level の取りうる値は後述する。残りの format 引き数は printf(3)と似たスタイルの書式とその書式に与える値である。format 中の 2 文字%m はその時点での errno に関連するエラーメッセージ文字列(strerror)によって置き換えられる。vsyslog()関数は syslog()と同じ機能を持つが、可変引き数リストを指定することができる点が異なる。指定された引き数は、stdarg(3)可変引き数リストマクロを用いて取得される。

引き数

ここでは option と facility と priority の値を設定するのに使用されるパラメータを説明する。

オプション

下記の値を OR したものが openlog()の option 引き数になる

- LOG_CONS
  エラーがあれば、システムロガーに送る一方でシステムコンソールにも直接書く。
- LOG_NDELAY
  ログ記録用プログラムとの接続を即座に開始する(通常は、最初のメッセージが記録される時に接続を開く)
- LOG_NOWAIT
  メッセージを記録する際に生成される子プロセスの終了を待たない。(GNU C ライブラリは子プロセスを生成しない。したがって、このオプションは Linux では無効である。)

- LOG_ODELAY

  LOG_NDELAY の反対。syslog()が呼ばれるまで、接続の開始を行わない。(このオプションはデフォルトであり、特に指定する必要はない。)
- LOG_PERROR

  stderr にも出力する。(SUSv3 では定義されていない)
- LOG_PID

  個々のメッセージに PID を含める。

ファシリティ

facility 引き数はメッセージに記録されるプログラムのタイプを指定するために使われる。これによって異なるタイプのプログラムからのメッセージは異なる扱いをするように設定ファイル(syslog.conf(5))に定義できる。

- LOG_AUTH

  セキュリティ/認証メッセージ(非推奨。代わりに LOG_AUTHPRIV を使用すること)
- LOG_AUTHPRIV

  セキュリティ/認証メッセージ(プライベート)
- LOG_CRON

  クロックデーモン(cron と at)
- LOG_DAEMON

  特定の facility 値を持たないシステムデーモン
- LOG_FTP

  ftp デーモン
- LOG_KERN

  カーネルメッセージ
- LOG_LOCAL0 から LOG_LOCAL7

  ローカルな使用のためにリザーブされている
- LOG_LPR

  ラインプリンタ・サブシステム
- LOG_MAIL

  メール・サブシステム
- LOG_NEWS

  USENET ニュース・サブシステム
- LOG_SYSLOG

  syslogd によって内部的に発行されるメッセージ
- LOG_USER(デフォルト)

  一般的なユーザレベルメッセージ

- LOG_UUCP
  UUCP サブシステム

レベル

これはメッセージの優先度を指定する。優先度の高いものから低いものの順で下記する。

- LOG_EMERG
  システムが使用不可
- LOG_ALERT
  直ちに行動を起こさなければならない
- LOG_CRIT
  危険な状態
- LOG_ERR
  エラーの状態
- LOG_WARNING
  ワーニングの状態
- LOG_NOTICE
  通常だが重要な状態
- LOG_INFO
  インフォメーションメッセージ
- LOG_DEBUG
  デバッグレベルのメッセージ

setlogmask(3)関数を使用して、指定されたレベルのメッセージだけを記録するように制限することができる。

準拠

openlog()、closelog()、syslog()は SUSv2 と POSIX 1003.1-2001 で規定されている。(但し syslog()は除く)POSIX 1003.1-2001 では facility として LOG_USER と LOG_LOCAL*の値が規定されているだけである。しかしながら、LOG_AUTHPRIV と LOG_FTP という例外はあるが、それ以外の facility の値は多くの Unix システムで使われている。option の値の LOG_PERROR の値は、POSIX 1003.1-2001 では規定されていないが、Unix の多くのバージョンで使用可能である。

履歴

syslog ファンクション・コールは BSD 4.2 から実装された。BSD 4.3 には openlog()、syslog()、closelog()、setlogmask()が実装されている。また、4.3BSD-Reno には vsyslog()が実装されている。当然ながら初期

の v*関数は<stdarg.h>とは互換性のない<varargs.h>の仕組を使用したものである。

注意

openlog()呼び出しの ident 引き数は、値がそのまま保持されていることを前提にしている。それゆえ、ident で指定された文字列が変更されると、syslog()は変更された文字列の追加するだろうし、指定された文字列が存在しなくなった場合、結果は未定義である。最も移植性がある方法は、文字列定数を使用することである。

ユーザーから与えられたデータを format として渡してはならない。代わりに syslog(priority, "%s", string); を使うこと。

関連項目

logger(1)、setlogmask(3)、syslog.conf(5)、syslogd(8)