A Handbook for Octave
Version 0.2β

2005 年 6 月 1 日

2

目次

第 1 章

文法編

1.1 入力の概略

基本的な入力規則は、以下の通りである。

- コマンド（命令語）や変数名などを入力して、改行する（Enter キーを押す）と、結果が画面に表示される。
- 入力すべき内容が完了していないときは、改行後も継続して入力する状態になる。
- コマンドが 1 行に書ききれないときには、行末に ... または \を置いて改行することで、継続して入力できる（継続行）。
- 結果を画面に表示しないときには、行末に ;（セミコロン）を付加する。
- コマンドには大文字と小文字の区別がある。
- #（シャープ文字）以降の入力内容は、すべて無視される（コメント）。
- 通常は入力履歴が保存され、↑ キーで呼び出すことができる。
- 関数名または変数名の入力途中で Tab キーを押すと、入力内容が補完される。

1.2 データ型と変数

1.2.1 組み込みデータ型

スカラ

実数は、数値のみか e を用いた指数式（10^x）で表す。複素数は、虚数単位 i を末尾に付けることで識別する。

例：105　　105、1.05e+2、1050e-1

例：$3+4i$　3+4i、3.0+4.0i、0.3e1+40e-1i

行列

全体を [と] でくくる。要素は、改行または; で行を区切る。転置するには、末尾に' をつける。以下の表現は、いずれも同じ値を意味する。

```
[1 2 3      [1 2 3; 4 5 6]     [1 4; 2 5; 3 6]'
 4 5 6]
```

範囲

始点:**増分**:**終点**の形で指定する。「始点」から「増分」きざみで「終点」までの値の集合を意味する。増分を省略すると 1 とみなす。

例：3:5　　意味：[3 4 5]

例：3:0.5:5　意味：[3.0 3.5 4.0 4.5 5.0]

文字列

シングルクォーテーション'、またはダブルクォーテーション"でくくる。どちらを用いても良いが、"の内部では特殊記号（エスケープシーケンス）が利用できる。

例：'Hello'　　意味：Hello という文字列

例："Thank you\n"　意味：Thank you の後に改行文字を含む文字列

文字列を要素とする行列も作成可能であるが、数値と混在できない。このとき文字列は列方向に連結されるので、列ベクトル（配列）となる。以下の行列は、すべて同じ値となる。

```
['abcdef'   ['abc' 'def'   ['abc' 'def'; 'gh' 'ij']
 'ghij']     'gh'  'ij']
```

構造体

構造体.**メンバ**は、スカラ、行列、文字列の各変数をメンバとする構造体である。メンバに構造体を含むことができる。

リスト

`list(a,b,...)` とすることで、`a,b,...` を要素とするリストを作成できる。

1.2.2 変数

変数には、組み込みデータ型の値を代入することができる。変数に値を代入するには、代入演算子=を用いる。Octave の変数には以下の特徴がある。

- 変数名はアルファベット大文字・小文字・数字・アンダーバーのみが使用できる。
- 頭文字は、数字以外でなければならない。
- 変数名の大文字と小文字は区別される。
- 変数名の長さには制限がない。
- 型が存在せず、宣言の必要もない。

変数の値は、関数内でのみ参照できる（ローカル変数）。いずれの関数からも参照可能にするには、`global` をつけて宣言しなければならない（グローバル変数）。

Octave の動作を規定する組み込み変数が存在する。ユーザは、必要に応じてその値を変えることができる。

1.3 表現

1.3.1 行列の添え字

行列（変数）の直後に (index) を追加することにより、任意の要素を参照することができる。index には、スカラ、ベクトル、範囲を指定できる。単に : だけを指定したとき、指定した行または列の全要素を意味する。

たとえば a=[1 2; 3 4] とすると、以下の表現はすべて「1 行めの全要素」を意味する。

```
a(1, [1 2])    a(1, 1:2)    a(1, :)
```

index には、同じ値を重複して指定することもできる。

ある変数にスカラが代入されているとき、添え字に 1 の行列を指定すると、その対応する要素にもとのスカラ値が代入される。たとえば a=13 のとき a([1 1], [1 1 1]) とすると 2×3 行列の要素がすべて 13 となる。

1.3.2 関数の呼び出し

関数の一般的な呼び出し書式は、以下のようになる。

$$[out1\ out2\ ...] = \mathbf{func}\ (arg1,\ arg2,\ ...)$$

適切な数の引数 *arg1, arg2, ...* をカンマで区切って与える。戻り値が複数のときは、[ ] で囲う。受け取る変数が規定よりも少ないとき、左から順に値を受け取る。変数を受け取らないとき、標準出力に出力する。

任意の関数を自作することができる。また、自作関数をファイルに記憶しておき、通常の関数と同様に呼び出すことができる。Octave で利用可能な関数には、実行形態がある。

組み込み関数 C や Frotran などの言語で記述され、コンパイル済みのバイナリとして実行する関数

動的リンク関数 C や Frotran などの言語で記述され、コンパイル済みのバイナリとして必要に応じて動的にリンクされるユーザ定義関数

関数ファイル Octave の命令語を記述したテキストファイル

マッピング関数 行列・ベクトルの要素どうしに作用する関数

1.3.3 算術演算子

x と y が行列とベクトルのとき、演算に対して整合性が必要である。

x `+` y	加算	x `\` y	左除算（掃き出し）
x `-` y	減算	x `^` y	ベキ乗
x `*` y	乗算	x `**` y	（同上）
x `/` y	除算	x`'`	転置
x `.+` y	要素ごと加算	x `.\` y	要素ごと左除算
x `.-` y	要素ごと減算	x `.^` y	要素ごとベキ乗
x `.*` y	要素ごと乗算	x `.**` y	（同上）
x `./` y	要素ごと除算	x`.'`	転置
`-`x	符号の反転	`+`x	明示的な正

1.3.4 比較演算子

x `<` y	x が y より小さいとき真
x `<=` y	x が y 以下のとき真
x `==` y	x が y と等しいとき真
x `>=` y	x が y 以上のとき真
x `>` y	x が y より大きいとき真
x `!=` y	x が y と等しくないとき真
x `~=` y	（同上）
x `<>` y	（同上）

1.3.5 論理演算子

bool1 `&` *bool2*	*bool1* と *bool2* の要素ごとの and
bool1 `|` *bool2*	*bool1* と *bool2* の要素ごとの or
`!`*bool*	*bool* の not
`~`*bool*	（同上）
bool1 `&&` *bool2*	`all(all(`*bool1*`))` と `all(all(`*bool2*`))` の and
bool1 `||` *bool2*	`all(all(`*bool1*`))` と `all(all(`*bool2*`))` の or

1.3.6 代入演算子

obj = *val*	右辺の *val* を左辺の *obj* に代入する
a=[10 12 15]	a に 1×3 のベクトルを代入
a(3,:)=-1	a の 3 行めの要素をすべて-1 とする
a(3,:)=[]	a の 3 行めの要素をすべて削除する
a=b=c=5	a、b および c に 5 を代入する

1.3.7 インクリメント演算子

++*x*	*x* を評価する前に値を 1 増やす
--*x*	*x* を評価する前に値を 1 減らす
x++	*x* を評価した後に値を 1 増やす
x--	*x* を評価する後に値を 1 減らす

1.3.8 演算子の順位

弱い	;, ,	ステートメントの区切り
⇑	=	右辺と左辺をグループ化する
	||, &&	論理 and と or
	|, &	要素ごとの and と or
	<,<=,==,>=,>,!=,~=,<>	関係演算子
	:	コロン（範囲）
	+, -	加算と減算
	, /, \, .\, ., ./	乗算と除算
	'. .'	転置
⇓	+, -, ++, --, !, ~	その他の演算子
強い	^, **, .^, .**	ベキ乗

1.4 ステートメント

1.4.1 if ステートメント

condition が真のとき、対象となるブロックを実行する。

```
if(condition)
    then-body
endif
```

```
if(condition)
    then-body
else
    else-body
endif
```

```
if(condition)
  then-body
elseif(condition)
  elseif-body
else
  else-body
endif
```

`if` のブロック内に、別の `if` ステートメントを置くことができる。

1.4.2 switch ステートメント

```
switch expression
  case label
    command_list
  case label
    command_list
  ...

  otherwise
    command_list
endswitch
```

expression の値に一致する *label* の *command_list* を実行する。一致しなかった場合は `otherwise` のブロックを実行する（省略可能）。*label* は、上から順に評価する。
command_list を実行後、`switch` を終了する（次のラベルには移動しない）。

1.4.3 while ステートメント

```
while(condition)
    body
endwhile
```

condition が真であるあいだ、*body* を繰り返し実行する。ループ先頭で *condition* を評価する。

1.4.4 do-until ステートメント

```
do
    body
until(condition)
```

condition が真であるあいだ、*body* を繰り返し実行する。ループの末端で *condition* を評価する。

1.4.5 for ステートメント

```
for var=expression
    body
endfor
```

変数 *var* に *expression* の値を代入できるあいだ、*body* を繰り返し実行する。*expression* には、範囲や行列が指定できる。

1.4.6 構造体の要素を得る for ステートメント

```
for [val, key]=expression
    body
endfor
```

構造体 *expression* からキー（変数名）と値を取り出す。すべての値を取り出すまで、処理を繰り返す。

1.4.7 break、continue ステートメント

```
break
continue
```

`break` は、現在の `while` または `for` のループを抜ける。`continue` は、現在のループの末端に移動する（先頭に戻る）。

1.4.8 unwind_protect ステートメント

```
unwind_protect
    body
unwind_protect_cleanup
    cleanup
end_unwind_protect
```

まず *body* を実行し、ここでエラーが発生したときは、*cleanup* ブロックを実行する。

1.4.9 try ステートメント

```
try
    body
catch
    cleanup
end_try_catch
```

`unwind_protect` と同様の処理をする。*body* ブロックでのエラーメッセージは表示しないが、組み込み変数 `__error_text__` に格納される。

入力に関しては、入力の概略（1 ページ）も参照するとよい。

1.5 関数ファイルとスクリプトファイル

1.5.1 関数の定義

関数定義の一般形は、以下のようになる。

```
function [ret-list] = name (arg-list)
    body
endfunction
```

関数名 *name* を定義する。引数をもつ場合は引数名を *arg-list* にカンマ区切りで記述する。返り値がある場合は、*arg-list* に記述する。*arg-list* と *ret-list* は、省略することができる。対話モードであっても、関数を定義することができる。

ユーザ定義関数を呼び出す方法は、Octave の標準関数と同様である。関数実行時、渡された引数の数は `nargin` 変数に、指定された戻り値の数は `nargout` 変数にセットされる。なお、引数は値渡しである。関数から抜けるには `return` コマンドを用いる。

1.5.2 可変長引数リスト

関数定義時、*arg-list* に `...` を記述する。この部分以降が可変長引数となる。引数を得る **va_arg ()**、内部ポインタを戻す **va_start ()** を用いる。

1.5.3 可変長戻り値リスト

関数定義時、*ret-list* に `...` を記述する。この部分以降が可変長戻り値となる。引数を順に返す **vr_val ()** を用いる。

1.5.4 関数ファイル

関数ファイルには、ファイル名と同じ名前の関数を 1 つだけ記述する。拡張子は `.m` としておく。関数ファイルは、組み込み変数 `LOADPATH` で指定したパスに置いておく。実行には、そのファイル名を入力する。

ファイル中の#または % から行末までは、コメントとして無視する。

1.5.5 スクリプトファイル

Octave のコマンドを記述する。関数ファイルと区別するため、`function` で開始してはいけない（先頭に `1;` などと記述しておく）。複数の関数を含めることができる。

ファイルの拡張子を `.m` とすると、関数ファイルと同様に実行できる。任意のスクリプトファイルを実行するには、**source** 関数を用いる。

1.6 起動オプション・定数・書式

1.6.1 起動オプション

`--debug` `-d`	デバッグモードに入る
`--echo-commands` `-x`	実行時にそのコマンドを表示する
`--execpath` *path*	実行プログラムの検索パスを指定する

`--help`
`-h`
`-?`
	短いヘルプメッセージを表示する

`--info-file` *filename*	使用する info file を指定する
`--info-program` *program*	使用する info program を指定する
`--interactive` `-i`	強制的に対話型にする
`--no-history` `-h`	コマンドラインヒストリを無効にする
`--no-init-file`	初期化ファイルを読まない
`--no-line-editing`	コマンドライン編集を無効にする
`--no-site-file`	site-wide ファイルを読まない
`--norc` `-f`	初期化ファイルをいっさい読まない
`--path` *path* `-p` *path*	関数ファイル検索パスを指定する

`--silent`
`--quiet`
`-q`
	起動メッセージを表示しない

`--traditional` `--braindead`	MATLAB 互換の設定で起動する
`--verbose` `-V`	冗長な出力を行う
`--version` `-v`	バージョン番号を表示する
file	*file* を実行する

1.6.2 プロンプトのカスタマイズ

組み込み変数 PS1、PS2 および PS4 に指定する識別子である。実際に指定するには、バックスラッシュを重ねる必要がある（例：\\t など）。

\t	時刻
\d	日付
\n	改行
\s	プログラム名（'octave'）
\w	現在の作業ディレクトリ
\W	現在の作業ディレクトリの base name
\u	現在のユーザ名
\h	ホスト名（最初の. までの名称）
\H	ホスト名
\#	起動からのコマンド数
\!	このコマンドのヒストリ数
\$	別ユーザなら'#'、実ユーザなら'$'
\nnn	8 進数コード nnn の文字
\\	バックスラッシュ

1.6.3 文字定数のエスケープシーケンス

\\	バックスラッシュ（\）
\"	ダブルクォーテーション（"）
\'	シングルクォーテーション（'）
\a	ベル（Ctrl-G）ASCII-code:7
\b	後退（Ctrl-H）ASCII-code:8
\f	改頁（Ctrl-L）ASCII-code:12
\n	改行（Ctrl-J）ASCII-code:10
\r	復帰（Ctrl-M）ASCII-code:13
\t	タブ（Ctrl-I）ASCII-code:9
\v	垂直タブ（Ctrl-K）ASCII-code:11

1.6.4 ワイルドカード

?	任意の 1 文字
*	任意の 0 文字以上の文字列
[list]	[] 内のいずれかの文字（[a-z] のように範囲指定も可能）

1.6.5 C 言語スタイルの入出力書式

基本的な書式は、以下のようになる。

出力時　% フラグ 出力幅 [. 精度] 種類 変換型
入力時　% フラグ 入力幅 種類 変換型

出力時、デフォルトの精度（小数点以下桁数）は 6 である。

出力型

%d	符号つき 10 進数	%e	浮動小数点（指数 e 形式）
%i	符号つき 10 進数	%E	浮動小数点（指数 E 形式）
%o	符号なし 8 進数	%g	場合に応じて %f と %e
%u	符号なし 10 進数	%G	場合に応じて %f と %E
%x	符号なし 16 進数（小文字）	%c	単一の文字
%X	符号なし 16 進数（大文字）	%s	文字列
%f	浮動小数点（小数点位置指定）	%%	% 文字

出力フラグ

-	左詰めにする
+	値が正のとき + 符号を付ける
	符号が付かないときは空白で埋める
#	%o のとき 0 を，%x、%X のとき 0x を先頭に置く
0	空白の代わりに 0 で埋める（-とは同時に指定できない）

入力型

%d	符号つき 10 進表記
%i	C 言語記述の符号つき 10 進表記
%o	符号なし 8 進表記
%u	符号なし 10 進表記
%x, %X	符号なし 16 表記
%e, %E, %f	浮動小数点表記
%g, %G	
%s	文字列（空白を含まない）
%c	1 文字以上の文字（幅による）
%%	% 文字

第 2 章
コマンドリファレンス編

凡例

function (*arg1, arg2, ...*)
　引数として *arg* をとり、単一のオブジェクト（変数，行列，文字列など）を戻り値にする関数である。

[*ret1, ret2*] = **function** (*arg, ...*)
　上記と同様だが，複数の戻り値をもつ関数である。戻り値の数よりも受け取る変数の数が少ない場合，左側の戻り値から返す。
　これら関数の引数部分で用いる記号は，以下のことを意味する。

- ... 可変数の引数を設定できる。
- † この引数を省略できる。

command *arg*
　Octave に組み込まれているコマンド（キーワード）である。

`builtin_variable`
　組み込み変数である。その多くは後半にまとめてあるが，とくに関数と共に用いるものについては，その付近に説明している。

2.1 起動・終了・全般

exit (*status*†)
quit (*status*†)
整数値 *status* を返して Octave を終了する。

atexit (*fcn*)
関数 *fcn* を呼び出して終了する。

help
ヘルプを表示する。

clc ()
home ()
画面内容をすべて消去する。

completion_matches (*hint*)
文字列 *hint* に対する補完を行う。

history *options*
入力履歴をすべて表示する。*options* が指定できる。

`-w` *file*	現在の履歴を *file* に書き込む
`-r` *file*	*file* から履歴を読み込む
n	最新の履歴を *n* 行だけ表示する
`-q`	履歴番号を表示しない

edit_history *opt1 opt2*
現在の履歴から *opt1* まで，または *opt1* から *opt2* までをエディタで編集する。

run_history *[first] [last]*
fiest から *last* までの履歴を実行する。

read_readline_init_file (*file*)
readline のライブラリ初期化ファイル *file* を読み込む。

diary *options*
入力した全コマンドとその実行結果をファイルに出力する。

`on`	今後の実行内容をファイル `diary` に出力開始する
`off`	ファイルへの出力を停止する
file	出力ファイル名を指定する

echo *options*

コマンド実行時の表示を操作する。

`on`	スクリプトファイル実行時にコマンドを表示する
`off`	スクリプトファイル実行時にコマンドを表示しない
`on all`	スクリプトファイルと関数の実行時にコマンドを表示する
`on all`	スクリプトファイルと関数の実行時にコマンドを表示しない

2.2 データのサイズと型

typeinfo (*expr*†)
　expr のデータ型を文字列で返す。指定しないときは全データ型を返す。

columns (*a*)
　a の列数を返す。

rows (*a*)
　a の行数を返す。

length (*a*)
　a の長さを返す。行列のときは行か列の大きい方を返す。

size (*a*, *n*)
　a の行数と列数を返す。*n* が 1 のとき行数，2 のとき列数のみ返す。

isempty (*a*)
　a が行数 0 かつ列数 0 のとき 1 を返す。

isnumeric (*x*)	*x* が数値のとき 0 以外の値を返す。
isreal (*x*)	*x* が実数値のとき true を返す。
is_complex (*x*)	*x* が複素数値のとき true を返す。
is_matrix (*a*)	*a* が行列のとき 1 を返す。
is_vector (*a*)	*a* がベクトルのとき 1 を返す。
is_scalar (*a*)	*a* がスカラのとき 1 を返す。
is_square (*x*)	*x* が正方行列のときその次数を返す。
is_symmetric (*x*, *tol*†)	誤差 *tol* で *x* が対称行列のときその次数を返す。
is_bool (*x*)	*x* がブール数のとき true を返す。

2.3 文字列

blanks (*n*)
n 個の空白からなる文字列を返す。

int2str (*n*)
num2str (*x*)
数値を文字列に変換する。

com2str (*zz, flg*†)
複素数 *zz* を *flg* の書式に沿って文字列に変換する。

setstr (*x*)
行列の要素を対応する ACII 文字列に変換する。

strcat (*s1, s2, ...*)
すべての引数を結合した文字列を返す。

str2mat (*s_1, ..., s_n*)
文字列 *s_1* から *s_n* を要素とする文字列行列を返す。

isstr (*a*)
a が文字列のとき 1 を返す。

deblank (*s*)
文字列 *s* の終端の空白を削除する。

findstr (*s, t, overlap*†)
文字列 *s* に *t* が出現するすべての位置を返す。*overlap* が 0 以外のとき，重複を許す。

index (*s, t*)
文字列 *s* に *t* が最初に現れる位置を返す。見つからないときは 0 を返す。

rindex (*s, t*)
文字列 *s* に *t* が最後に現れる位置を返す。見つからないときは 0 を返す。

split (*s, t*)
文字列 *s* を区切り文字列 *t* で分割した文字列ベクトルを返す。

strcmp (*s1, s2*)
ふたつの文字列が等しいとき 1 を返す（C 言語と逆）。

strrep (*s*, *x*, *y*)
　文字列 *s* 中の文字列 *x* を *y* で置き換える。

substr (*s*, *beg* *len*†)
　文字列 *s* 中の *beg* 文字めから *len* 文字分を取り出す。

bin2dec (*s*)
　2 進数文字列 *s* を数値に変換する。

dec2bin (*n*)
　正の数値 *n* を 2 進数文字列に変換する。

dec2hex (*n*)
　正の数値 *n* を 16 進数文字列に変換する。

hex2dec (*s*)
　16 進数文字列 *s* を数値に変換する。

str2num (*s*)
　文字列 *s* を数値に変換する。

toascii (*s*)
　文字列 *s* を ASCII コードの配列に変換する。

tolower (*s*)
　文字列 *s* 内の大文字を小文字に変換する。

toupper (*s*)
　文字列 *s* 内の小文字を大文字に変換する。

do_string_escapes (*strings*)
　文字列 *string* 中の特殊文字をエスケープ形式に変換する。

undo_string_escapes (*s*)
　文字列 *s* 中のエスケープ形式を特殊文字に変換する。

isalnum (s)　s がアルファベットか数字のとき 1 を返す。
isalpha (s)　s がアルファベットのとき 1 を返す。
isascii (s)　s が ASCII 文字（0 から 127 の数値）のとき 1 を返す。
iscntrl (s)　s が制御文字のとき 1 を返す。
isdigit (s)　s が 10 進数値のとき 1 を返す。
isgraph (s)　s が印刷可能文字（空白以外）のとき 1 を返す。
islower (s)　s が小文字のとき 1 を返す。
isprint (s)　s が印刷可能文字（空白含む）のとき 1 を返す。
ispunct (s)　s が句読点のとき 1 を返す。
isspace (s)　s が空白文字（改行やタブ等含む）のとき 1 を返す。
isupper (s)　s が大文字のとき 1 を返す。
isxdigit (s)　s が 16 進数値のとき 1 を返す。

注）これらに文字列を渡すと，各々の文字について真偽を返す。

2.4 構造体

is_struct (*expr*)
　expr が構造体のとき 1 を返す。

struct_contains (*expr*, *name*)
　expr が構造体であり *name* の要素を含むとき 1 を返す。

struct_elements (*struct*)
　構造体 *struct* の要素名を文字列で返す。

2.5 コンテナ

list (*a1*, *a2*, ...)
　引数で指定したリストを作成する。

nth (*list*, *n*)
　リスト *list* の *n* 番めの要素を返す。

append (*list*, *a1*, *a2*, ...)
　リスト *list* に指定した要素を追加した新たなリストを返す。

reverse (*list*)
　リスト *list* を逆順にした新たなリストを返す。

splice (*list_1*, *offset*, *length*†, *list_2*†)
　リスト *list_1* の *offset* 番めから *length* 個の要素を *list_2* と置き換える。

cell (*x*)
cell (*n*, *m*)
　サイズ $x \times x$ あるいは $n \times m$ の新たなセル配列を作成する。

iscell (*x*)
　x がセル配列のとき true を返す。

2.6 I/O ストリーム

is_stream (*x*)
　x が I/O ストリームのとき true を返す。

2.7 変数

is_global (*name*)
name がグローバル変数のとき 1 を返す。

clear *[-x] pattern ...*
ワイルドカード *pattern* にマッチした名前をシンボルテーブルから削除する。*-x* を指定すると，マッチしないものを削除する。

who *options pattern ...*
whos *options pattern ...*
ワイルドカード *pattern* にマッチするシンボルを一覧表示する。*options* には以下のオプションが指定できる。オプションは，複数指定できる。

`-all`	現在定義されているすべてのシンボルを表示
`-builtins`	すべての組み込み変数および関数を表示
`-functions`	ユーザ定義関数を表示
`-long`	データ型とサイズなどを含めた詳細な情報を表示
`-variables`	ユーザ定義変数を表示

関数 **whos** は，**who** *-long* と等しい。

exist (*name*)
name が変数として存在するなら 1，パス内に関数ファイル（拡張子`.m`）として存在するなら 2，パス内に`.oct` ファイルとして存在するなら 3，組み込み関数なら 5，組み込み定数なら 6 を返す。

document (*symbol, text*)
symbol に注釈文字列 *text* をつける。

type *options name ...*
name が指す関数の定義と種類（ユーザ定義か組み込みか）を表示する。オプション`-q` は，関数の種類を表示しない。

which *name ...*
name の型を表示する。関数ファイルの場合は所在も表示する。

2.8 評価

eval (*try*, *catch*)
　文字列 *try* を文として実行し，その結果を返す。失敗した場合は文字列 *catch* を実行する。

feval (*name*, ...)
　name で指定した関数を実行する。関数の引数は，続けて記述する。

2.9 関数とスクリプトファイル

`nargin`
関数に渡された引数の数。関数が呼ばれるたびに自動的に更新される。

`nargout`
関数が返す値の数。関数が呼ばれるたびに自動的に更新される。

nargchk (*nargin_min*, *nargin_max*, *n*)
n が *nargin_min* から *nargin_max* の範囲になければメッセージを表示する。

va_arg ()
次の引数を返し，内部ポインタを進める。返す値がなくなればエラーを返す。

va_start ()
内部ポインタを最初の引数に戻す。

all_va_args
オプショナル引数のリストを表す。

vr_val (*x*)
可変数の戻り値をもつ関数で，順次，戻り値を返す。

return
関数またはスクリプトファイルを抜け，呼び出し元に戻る。

rehash ()
`LOADPATH` のディレクトリキャッシュを再初期化する。

file_in_loadpath (*name*)
name を `LOADPATH` から検索する。

source (*file*)
file を読み込んで実行する（関数名とファイル名が異なってもよい）。

2.10　エラー処理

error (*template*, ...)
　テンプレート *template* の内容を `error:` に続けて表示する。

warning (*msg*)
　警告文として *msg* を `warning:` に続けて表示する。

usage (*msg*)
　使用法として *msg* を `usage:` に続けて表示する

perror (*name*, *num*)
　関数名 *name* のエラー番号 *num* に対応するメッセージを表示する。

strerror (*name*, *num*)
　関数名 *name* のエラー番号 *num* に対応する文字列を返す。

2.11 入出力

2.11.1 入出力全般

more
more *on*
more *off*
ページ単位表示の切り替えをする。

fflush (*fid*)
fid への出力をフラッシュする。

2.11.2 基本的な入出力

標準出力

ans
最新の計算結果を保持する。

fdisp (*fid*, *x*)
システム出力 *fid* に値 *x* を出力し，改行する。

disp (*x*)
値 *x* を出力し，改行する。

format *options*
disp による出力と Octave 標準出力のフォーマットを指定する。

short	最大 8 文字幅で有効桁数 3 桁とする
long	最大 24 文字幅で有効桁数 15 桁とする
short e	e を用いた short の表示
long e	e を用いた long の表示
short E	E を用いた short の表示
long E	E を用いた long の表示
free	小数点をそろえないフリーフォーマット
none	小数点をそろえる表示とする
bank	小数点の右側を固定する
+	空行列の要素に + を表示する
hex	メモリ格納順に 16 進数で表示
bit	メモリ格納順に 2 進数で表示

標準入力

input (*prompt*)
input (*prompt*, `'s'`)
　prompt を表示してキー入力を待ち，入力値を返す。`s` を指定すると文字列として読み込む。

menu (*title*, *opt1*, ...)
　title をタイトルとし，*opt1*, ... を項目としたメニューを表示し，キー入力を待つ。選択した項目番号を返す。

keyboard (*prompt*†)
　prompt をプロンプトとしてデバッグのための入力を行う。

kbhit (*x*†)
　キーボードから 1 文字入力し，値を返す。引数 *x* を付けると，キー入力と同時に終了する。

ファイル入出力

save *options*, *file*, *v1 v2* ...
　変数 *v1 v2* ...（指定しないときは全変数）をファイル *file* に保存する。ファイル名を`-`とすると端末に表示する。保存する変数名には，ワイルドカードが使用可能である。

`-ascii`	Octave テキスト形式で保存
`-binary`	Octave バイナリ形式で保存
`-float-binary`	Octave バイナリ（単精度）形式で保存
`-mat-binary`	MATLAB バイナリ形式で保存
`-mat4-binary`	MATLAB v.4 バイナリ形式で保存
`-hdf5`	HDF5 形式で保存
`-float-hdf5`	HDF5（単精度）形式で保存
`-save-builtins`	組み込み変数も保存

load *options*, *file*, *v1 v2* ...
　変数 *v1 v2* ...（指定しないときは全変数）をファイル *file* から読み込む。変数名には，ワイルドカードが使用可能である。

`-force`	現在の変数を読み込んだ内容で上書きする
`-ascii`	Octave テキスト形式で読み込む
`-binary`	Octave バイナリ形式で読み込む
`-float-binary`	Octave バイナリ（単精度）形式で読み込む
`-mat-binary`	MATLAB バイナリ形式で読み込む
`-mat4-binary`	MATLAB v.4 バイナリ形式で読み込む
`-hdf5`	HDF5 形式で読み込む
`-float-hdf5`	HDF5（単精度）形式で読み込む
`-import`	HDF5 ファイルをインポートする

2.11.3 C 言語スタイルの入出力

オープン・クローズと入出力

`stdin`
標準入力（ファイル id は 0）を表す。

`stdout`
標準出力（ファイル id は 1）を表す。

`stderr`
標準エラー出力（ファイル id は 2）を表す。

[*fid*, *msg*] = **fopen** (*name*, *mode*, *arch*†)
fidlist = **fopen** (`'all'`)
file = **fopen** (*fid*)

ファイル *name* をモード *mode* を *arch* 形式で開き，ファイル id の *fid* を返す。エラー発生時は id に −1 と *msg* を返す。オプション`'all'` は，開いている全 id を返す。オプション *fid* は，対応するファイル名を返す。指定できる *mode* は次の通りである。

`'r'`	読み出し専用で開く
`'w'`	書き込み専用で開く
`'a'`	追加書き込みで開く
`'r+'`	存在しているファイルを読み書き両用で開く
`'w+'`	読み書き両用で開く
`'a+'`	読み書き両用，追加書き込みで開く

指定できる *arch* は次の通りである。

`native`	このマシンのアーキテクチャに従う
`ieee-le`	IEEE リトルエンディアン
`ieee-be`	IEEE ビッグエンディアン
`vaxd`	VAX D 浮動小数点
`vaxg`	VAX G 浮動小数点
`cray`	Cray 浮動小数点

fclose (*fid*)
　id のファイルを閉じる。成功すると 1 を返す。

fputs (*fid, string*)
　ファイル *fid* に文字列 *string* を書き込む。

puts (*string*)
　文字列 *string* を標準出力に書き込む。

fgetl (*fid, len*)
　ファイル *fid* から（行端まで）*len* 文字読み込む。改行を除く。

fgets (*fid, len*)
　ファイル *fid* から（行端まで）*len* 文字読み込む。改行を含む。

書式付き入出力

　書式 *template* は、13 ページを参照のこと。

printf (*template, ...*)
　書式 *template* に従って `stdout` に出力する。

fprintf (*fid, template, ...*)
　書式 *template* に従って *fid* に出力する。

sprintf (*template, ...*)
　書式 *template* による文字列を返す。

[*val, count*] = **fscanf** (*fid, template, size*†)
[*v1, v2, ..., count*] = **fscanf** (*fid, template,* `'C'`)
　ファイル *fid* から書式 *templete* に従って読み込んだ値を返す。*count* に読み込んだ変数の数を返す。*size* には次のオプションが指定できる。

`Inf`	可能な限り読み込み，列ベクトルを返す
`nr`	`nr` 要素まで読み込み，列ベクトルを返す
`[nr, Inf]`	可能な限り読み込み，`nr` 行の行列を返す
`[nr, nc]`	`nr`×`nc` 要素を読み込み，`nr` 行の行列を返す

[*val, count*] = **sscanf** (*string, template, size*†)
[*v1, v2, ..., count*] = **sscanf** (*string, template,* `'C'`)
　文字列 *string* から書式 *templete* に従って読み込んだ値を返す。*count* に読み込んだ変数の数を返す。*size* は `fscanf` と同様である。

バイナリ入出力

[*val*, *count*] = **fread** (*fid*, *size*†, *precision*†, *skip*†, *arch*†)

ファイル *fid* から精度 *precision* でバイナリを読み込む。*precision* には以下の文字列を指定する。

`'char' 'char*1' 'integer*1' 'int8'`	単一の文字型
`'signed char' 'schar'`	符号つき文字型
`'unsigned char' 'uchar'`	符号なし文字型（デフォルト）
`'short'`	短整数型
`'unsigned short' 'ushort'`	符号なし短整数型
`'int'`	整数型
`'unsigned int' 'uint'`	符号なし整数型
`'long'`	長整数型
`'unsigned long' 'ulong'`	符号なし長整数型
`'float' 'float32' 'real*4'`	単精度浮動小数点
`'double' 'float64' 'real*8'`	倍精度浮動小数点
`'integer*2' 'int16'`	2 バイト整数型
`'integer*4' 'int32'`	4 バイト整数型

size は `fscanf` と同様である。*skip* は読み飛ばすバイト数を指定する。*arch* は `fopen` と同様である。

count = **fwrite** (*fid*, *data*, *precision*†, *skip*†, *arch*†)

ファイル *fid* に精度 *precision* でバイナリを書き込む。そのほかは `sscanf` と同様である。

その他のファイル関連処理

tmpnam ()

一意なテンポラリファイル名の文字列を返す。

feof (*fid*)

ファイル *fid* が終端であれば 1 を返す。

ferror (*fid*)

ファイル *fid* がエラー状態であれば 1 を返す。

freport ()

現在，開いているすべてのファイル状態を表示する。

ftell (*fid*)

ファイル *fid* の読み込み位置を文字数で返す。

fseek (*fid*, *offset*, *origin*†)

ファイル *fid* の位置を（*origin* から）*offset* 文字めに設定する。

`SEEK_SET`
ファイルの先頭（`fseek` の *origin* で用いる）

`SEEK_CUR`
ファイルの現在位置（`fseek` の *origin* で用いる）

`SEEK_END`
ファイルの終端（`fseek` の *origin* で用いる）

frewind (*fid*)
ファイル *fid* の位置を先頭に移動する。成功すると 1 を返す。

2.12 プロット

Octave では，外部プログラム gnuplot でプロットを実行する。プロットの詳細は gnuplot のマニュアルを参考のこと。

2.12.1 2 次元プロット

gplot *renges expression using title style*
expression の 2 次元グラフを表示する。オプションは以下の通り。

`range`	軸の範囲を [x 軸下端:x 軸上端] [y 軸下端:y 軸上端] で指定
`using`	*expression* にふたつ以上の列があるとき使用する列を指定
`title`	凡例に表示するタイトル
`style`	プロットのスタイル（ドット・ラインなど）を指定

gset *options*
gshow *options*
replot *options*
gnuplot のコマンド（`set`，`show`，`replot`）を実行する。

plot (*args*)
さまざまな *args* を与えて 2 次元グラフを表示する。たとえば，以下のような記述が可能。

`plot(y)`	y の内容を列ごとにライン表示
`plot(x,y,fmt)`	x 軸に x，y 軸に y の内容を `fmt` で打点
`plot(x1,y1,fmt1,x2,y2,fmt2)`	複数の系列をプロットすることも可能

与える *args* は，次のように解釈する。

- 引数が単一データのとき，x には要素番号，y には与えた値を設定する。
- 第 1 引数がベクトル，第 2 引数が行列のとき，x にベクトルの値，y には対応する行列の値を設定する。
- 第 1 引数が行列，第 2 引数がベクトルのとき，y にベクトルの値，x には対応する行列の値を設定する（上記の逆）。
- 2 つの引数がベクトルのとき，第 1 引数を x，第 2 引数を y とする。
- 2 つの引数が行列のとき，y の列が x の列に対応する。2 つの行列のサイズは，同じでなければならない。

書式 *fmt* として，以下のオプションが指定できる。これらは組み合わせることが可能である。

'-'	ライン（デフォルト）
'.'	打点（小さな点）
'@'	打点（記号類）
'-@'	ラインと打点（記号類）
'^'	インパルス
'L'	階段状
'#'	棒グラフ
'~'	誤差棒
'#~'	棒グラフと誤差棒
'n'	色の指定（n は 1 から 9 までの数値）
'nm'	色と打点記号の指定（m は 1 から 9 までの数値）
'+' '*' 'o' 'x'	それぞれの記号で打点
';str;'	*str* を凡例タイトルに表示する

hold (*args*†)
arg に `on` を指定すると，現在のプロットをそのままにして，新たなプロットを行う。`off` にすると，プロットのたびに上書きする。

ishold ()
`hold` が `on` であれば 1 を返す。

clearplot ()
clg ()
プロットウインドウの内容を消去する。

shg ()
プロットウインドウを表示する。`replot` と同じ動作である。

closeplot ()
プロットウインドウを閉じる。

purge_tmp_files ()
プロット時に作成したテンポラリファイルを削除する。

axis (*limits*, *option*†)
軸の範囲をベクトル *limits* で指定する。*limits* の最初の 2 つの要素は x 軸の上限と下限，次の 2 つの要素は y 軸，最後の 2 つは z 軸についての範囲とする。以下の *option* が指定できる。

`'square'`	縦横比を固定する
`'equal'`	x の距離を y の距離と等しくする
`'normal'`	バランスを戻す
`'auto'`	軸をデータに合わせて自動設定する
`'manual'`	現在の設定を固定する
`'tight'`	データの限界に固定する（未実装）
`'on'`	軸に tic-mark とラベルを表示する
`'off'`	軸に tic-mark とラベルを表示する
`'tic[xyz]'`	特定の軸に tic-mark を表示する
`'label[xyz]'`	特定の軸にラベルを表示する
`'nolabel'`	すべての軸にラベルを表示しない
`'ij'`	y 軸を逆にする
`'xy'`	y 軸を元に戻す

2.12.2 特殊な 2 次元プロット

bar (x, $y^{\dagger}$)
ベクトル x と y から棒グラフを描画する。

contour (z, n, x, y)
z の 3 次元等高線を描画する。

hist (y, $x^{\dagger}$, $norm^{\dagger}$)
x 個の区分で, y のヒストグラムを描画する。bar の総和を $norm$ とする。

loglog ($args$)
x と y の両軸を対数スケールとする。$args$ は `plot` と同じである。

polar ($theta$, rho, $fmt^{\dagger}$)
$theta$ と rho で表された極座標を描画する。fmt は線の種類とする。

semilogx ($args$)
x 軸を対数スケールとする。$args$ は `plot` と同じである。

semilogy ($args$)
y 軸を対数スケールとする。$args$ は `plot` と同じである。

stairs (x, $y^{\dagger}$)
ベクトル x と y から階段グラフを描画する。

2.12.3　3 次元プロット

gsplot *renges expression using title style*
　expression の 3 次元グラフを表示する。オプションは以下の通り。

`range`	軸を [x 軸下端:上端] [y 軸下端:上端] [z 軸下端:上端] で指定
`using`	*expression* にふたつ以上の列があるとき使用する列を指定
`title`	凡例に表示するタイトル
`style`	プロットのスタイル（ドット・ラインなど）を指定

mesh (*x, y, z*)
　`meshdom` による行列 *x* と *y* に対応する行列 *z* をメッシュ状に描画する。*x* と *y* がベクトルならば，$(x(j), y(i), z(i, j))$ を描画する。

[*xx, yy*] = **meshgrid** (*x, y*)
[*xx, yy*] = **meshgrid** (*x*)
　ベクトル *x* と *y* からメッシュに対応する行列 *xx* と *yy* を生成する。

[*xx, yy*] = **meshdom** (*x*)
　ベクトル *x* と *y* からメッシュに対応する行列 *xx* と *yy* を生成する。

2.12.4　その他のプロット関連

grid (*arg*)
　グリッドの描画を切り替える。*arg* には `on`（表示）か `off`（非表示）を指定する。

title (*string*)
　タイトルを *string* に設定する。

xlabel (*string*)
ylabel (*string*)
zlabel (*string*)
　x，*y* および *z* 軸ラベルを *string* に設定する。

2.12.5　多重プロット

mplot (*x, y*)
mplot (*x, y, fmt*)
mplot (*x1, y1, x2, y2*)

多重プロットが可能な `plot` である。

multiplot (*xn, yn*)
x と y 軸について *xn* と *yn* のサブプロット領域をもつ多重プロットを準備する。引数がゼロのとき複数プロットを終了する。

oneplot ()
多重プロットのとき，通常のプロットに戻る。

plot_border (...)
境界線の表示を設定する。以下のオプションを複数設定可能である。

`'blank'`	すべての境界線を表示しない
`'all'`	すべての境界線を表示する
`'north'`	上側の境界線を表示
`'south'`	下側の境界線を表示
`'east'`	右側の境界線を表示
`'west'`	左側の境界線を表示

subplot (*rows, cols, index*)
subplot (*rcn*)
多重プロットモードのとき，*rows* 行と *cols* 列のサブプロット領域の *index* に描画位置を設定する。*index* は 1 行めから列方向に番号付けする。

subwindow (*xn, yn*)
多重プロットモードのとき，次のプロットを行うサブプロット領域を指定する。

top_title (*string*)
bottom_title (*string*)
プロットのタイトル *string* を上部（下部）に表示する。

figure (*n*)
現在のプロットウインドウを *n* に指定する（X11 でのみ利用可能)。

2.13　行列演算

以下の解説について，特に明示しない限り，行列とベクトルを同一視する。

2.13.1　要素の検索と状態

any (x)
　ベクトル x の各要素について，ゼロでないとき 1 を返す。x が行列のとき，各行について，ゼロでない要素があるとき 1 を返す。

all (x)
　ベクトル x のすべての要素について，ゼロでないとき 1 を返す。x が行列のとき，各列について，すべての要素ゼロでないとき 1 を返す。

xor $(x,\ y)$
　ベクトル x と y の各要素について xor をとる。

is_duplicate_entry (x)
　x の重複していない要素数を返す。

diff $(x,\ k^{\dagger})$
　長さ n ベクトル x について，$x_2 - x_1, \ldots, x_n - x_{n-1}$ を返す。x が行列のとき，各行の差の行列を返す。k は差をとる回数を指定する。

isinf (x)
　x の各要素について，`Inf` のとき 1 を返す。

isnan (x)
　x の各要素について，`NaN` のとき 1 を返す。

finite (x)
　x の各要素について，有限値のとき 1 を返す。

find (x)
$[i,\ j] =$ **find** (x)
$[i,\ j,\ v] =$ **find** (x)
　x の各要素について，ゼロでない要素の index 番号のベクトルを返す。返り値を複数設定すると，i にはゼロでない要素の行番号，j には列番号，v にはその要素の値を返す。

$[err,\ y1,\ \ldots] =$ **common_size** $(x1,\ \ldots)$
　$x1,\ \ldots$ が同じサイズかスカラのときは，err に 0，それ以外は 1 を返す。

このとき *x1, ...* が行列のときは，入力値をそのまま *y1, ...* に返すが，スカラのときは，共通サイズの行列（要素はすべてスカラの値）を返す。

2.13.2 行列の変形

fliplr (x)
x の列について対称な行列を返す。

flipud (x)
x の行について対称な行列を返す。

rot90 $(x,\ n^{\dagger})$
行列 x を反時計回りに n 回だけ回転した値させる。

reshape $(a,\ m,\ n)$
行列 a を m 行 n 列に変形した行列を返す。

shift $(x,\ b)$
ベクトル x の各要素を b 回だけ循環シフトする。x が行列のとき，行単位でシフトする。

$[s,\ i] = \mathbf{sort}\ (x)$
行列 x を列単位でソートした行列 s を返す。i は s に対応する，もとの行列の index の行列である。

tril $(a,\ k^{\dagger})$
triu $(a,\ k^{\dagger})$
行列 a の下三角（`tril`）または上三角（`triu`）部分を返す。k は対角から何列まで返すかを指定する。

vec (x)
行列 x を列単位で連結した列ベクトルを返す。

vech (x)
正方行列 x の下三角を列単位で連結した列ベクトルを返す。

prepad $(x,\ l,\ c^{\dagger})$
postpad $(x,\ l,\ c^{\dagger})$
ベクトル x の先頭（`prepad`）または終端（`postpad`）に，値 c を長さ l になるまで追加する。x が行列のとき，行単位で追加する。

2.13.3 特別な行列

eye (*x*)
eye (*n*,*m*)
　サイズ *x* の単位行列を返す。あるいは単位行列の *n* 行 *m* 列部分を返す。

ones (*x*)
ones (*n*,*m*)
　要素がすべて 1 でサイズ *x* の正方行列，あるいは *n* 行 *m* 列の行列を返す。

zeros (*x*)
zeros (*n*, *m*)
　サイズ *x*，あるいは *n* 行 *m* 列のゼロ行列を返す。

rand (*x*)
rand (*n*, *m*)
rand (`'seed'`, *x*)
　要素が $(0, 1)$ の一様乱数でサイズ *x* の正方行列，あるいは *n* 行 *m* 列の行列を返す。`'seed'` 指定時，シードをスカラ *x* とする。*x* を省略すると，現在のシードを返す。

randn (*x*)
randn (*n*, *m*)
randn (`'seed'`, *x*)
　要素が $N(0, 1)$ の標準正規乱数を返す。そのほかは `rand` と同様である。

randperm (*n*)
　1 から *n* までの順序乱数のベクトルを返す。

diag (*v*, *k*[†])
　ベクトル *v* の要素を対角にもつ対角行列を返す。*k* 段の上（下）三角行列とする。*v* が行列のとき，その対角要素のベクトルを返す。*k* 段の上（下）三角行列を考慮する。

linspace (*base*, *limit*, *n*)
　base から *limit* までを *n* 等分した値を要素にする行ベクトルを返す。

logspace (*base*, *limit*, *n*)
　10^{base} から 10^{limit} までを *n* 等分した値を要素にする行ベクトルを返す。*limit* が π のとき，10^{base} から π までを範囲とする。

2.13.4 有名な行列

hankel $(c,\ r)$
最初の列が c で最後の行を r とする Hankel 行列を返す。

hilb (n)
次数 n の Hilbert 行列を返す。

invhilb (n)
次数 n の逆 Hilbert 行列を返す。

sylvester_matrix (k)
次数 $n = 2^k$ の Sylvester matrix を返す。

toeplitz $(c,\ r^\dagger)$
最初の列が c で最初の行を r とする Toeplitz 行列を返す。

vander (c)
最後の列が c である Vandermonde 行列を返す。

2.14 算術演算

2.14.1 ユーティリティ

ceil (x)
　x 以上で最小の整数を返す。複素数は `ceil(real(x))+ceil(imag(x))*I` を返す。

exp (x)
　e^x を返す（e は自然対数の底）。

fix (x)
　x の小数点以下を切り捨てる。複素数は `fix(real(x))+fix(imag(x))*I` を返す。

floor (x)
　x を超えない最大の整数を返す。複素数は `floor(real(x)) + floor(imag(x))*I` を返す。

gcd $(x, \ldots)$
　最大公約数を返す。引数はベクトルでもよい。

lcm $(x, \ldots)$
　最小公倍数を返す。引数はベクトルでもよい。

log (x)
　x の自然対数を返す。

log10 (x)
　x の常用対数を返す。

log2 (x)
$[f, e] =$ **log2** (x)
　x の底を 2 とする対数を返す。返り値は $\frac{1}{2} \leq |f| < 1$ と $x = f \cdot 2^e$ である。

nextpow2 (x)
　スカラ x の $2^n \geq |x|$ となる最初の整数 n を返す。ベクトル x について `nextpow2(length(x))` を返す。

pow2 (x)
pow2 (f, e)
　2^x または $f \cdot 2^e$ を返す。

rem (x, y)
x/y のあまりを返す。`x-y.*fix(x./y)` を計算する。

round (x)
x を四捨五入した値を返す。複素数は `round(real(x))+round(imag(x))*I` を返す。

sign (x)
x が正のとき 1，0 のとき 0，負のとき -1 を返す。複素数は `x./abs(x)` を返す。

sqrt (x)
x の平方根を返す。負のときは複素数を返す。

2.14.2 複素数演算

abs (z)
z の絶対値を返す。$|z| = \sqrt{x^2 + y^2}$ である。

arg (z)
angle (z)
z の偏角を返す。$\theta = \tan^{-1}(y/x)$ である。

conj (z)
z の共役複素数を返す。

imag (z)
z の虚数部分を実数として返す。

real (z)
z の実数部分を返す。

2.14.3 三角関数

sin (x)　　x の各要素の正弦（sin）を返す
cos (x)　　x の各要素の余弦（cos）を返す
tan (x)　　x の各要素の正接（tan）を返す
sec (x)　　x の各要素の正割（sec）を返す
csc (x)　　x の各要素の余割（cosec）を返す
cot (x)　　x の各要素の余接（cot）を返す
asin (x)　　x の各要素の逆正弦（arcsin）を返す
acos (x)　　x の各要素の逆余弦（arccos）を返す
atan (x)　　x の各要素の逆正接（arctan）を返す
asec (x)　　x の各要素の逆正割（arcsec）を返す
acsc (x)　　x の各要素の逆余割（arccosec）を返す
acot (x)　　x の各要素の逆余接（arccot）を返す
sinh (x)　　x の各要素の双曲線正弦（sinh）を返す
cosh (x)　　x の各要素の双曲線余弦（cosh）を返す
tanh (x)　　x の各要素の双曲線正接（tanh）を返す
sech (x)　　x の各要素の双曲線正割（sech）を返す
csch (x)　　x の各要素の双曲線余割（cosech）を返す
coth (x)　　x の各要素の双曲線余接（coth）を返す
asinh (x)　　x の各要素の逆双曲線正弦（asinh）を返す
acosh (x)　　x の各要素の逆双曲線余弦（acosh）を返す
atanh (x)　　x の各要素の逆双曲線正接（atanh）を返す
asech (x)　　x の各要素の逆双曲線正割（asech）を返す
acsch (x)　　x の各要素の逆双曲線余割（acosech）を返す
acoth (x)　　x の各要素の逆双曲線余接（acoth）を返す
atan2 (y, x)　　y, x の対応する要素の $\mathrm{atan}(y/x)$ を返す

2.14.4 和と積

sum (x)
　x の要素の総和を返す。

prod (x)
　x の要素の総積を返す。

cumsum (x)
　x の要素の累積和を返す。

cumprod (*x*)
x の要素の累積積を返す。

sumsq (*x*)
x の要素の平方和を返す。

2.14.5 特殊な関数

[*j*, *ierr*] = **besselj** (*alpha*, *x*, *opt*†)
[*y*, *ierr*] = **bessely** (*alpha*, *x*, *opt*†)
[*i*, *ierr*] = **besseli** (*alpha*, *x*, *opt*†)
[*k*, *ierr*] = **besselk** (*alpha*, *x*, *opt*†)
[*h*, *ierr*] = **besselh** (*alpha*, *k*, *x*, *opt*†)
さまざまな Bessel あるいは Hankel 関数を計算する。

`besselj`	第 1 種の Bessel 関数
`bessely`	第 2 種の Bessel 関数
`besseli`	第 1 種の修正 Bessel 関数
`besselk`	第 2 種の修正 Bessel 関数
`besselh`	第 1 種（*k*=1）あるいは第 2 種（*k*=2）の Hankel 関数

エラーコード *ierr* は次のようである。

`0`	正常終了
`1`	入力エラー；`NaN` を返す
`2`	オーバーフロー；`Inf` を返す
`3`	計算精度の減少により有効性が失われた
`4`	有効性が完全に失われた；`NaN` を返す
`5`	計算ができなかった；`NaN` を返す

[*a*, *ierr*] = **airy** (*k*, *z*, *opt*†)
第 1 種あるいは第 2 種の Airy 関数とその導関数を計算する。

`Ai(z)`	(*k*=0) Scale factor：`exp((2/3)*z*sqrt(z))`（デフォルト）
`dAi(z)/dz`	(*k*=1) Scale factor：`exp((2/3)*z*sqrt(z))`
`Bi(z)`	(*k*=2) Scale factor：`exp(-abs(real((2/3)*z*sqrt(z))))`
`dBi(z)/dz`	(*k*=3) Scale factor：`exp(-abs(real((2/3)*z*sqrt(z))))`

ierr は `bessel` と同様である。

beta (*a*, *b*)
ベータ関数を計算する。

betainc (*x*, *a*, *b*)
不完全ベータ関数を計算する。

bincoeff $(n,\ k)$
　2 項係数を計算する。

erf (z)
　誤差関数を計算する。

erfc (z)
　相補誤差関数を計算する。

erfinv (z)
　逆誤差関数を計算する。

gamma (z)
　ガンマ関数を計算する。

gammainc $(x,\ a)$
　不完全ガンマ関数を計算する。

lgamma $(a,\ x)$
gammaln $(a,\ x)$
　ガンマ関数の自然対数を計算する。

cross $(x,\ y)$
　3 次元ベクトルの外積を計算する。

commutation_matrix $(m,\ n)$
　交換行列 $K_{m,n}$ を計算する。

duplication_matrix (n)
　duplication matrix D_n を計算する。

2.14.6　定数

`I, J, i, j`
　虚数（$\sqrt{-1}$）を表す。`I` と `J` は上書きできない。

`Inf, inf`
　無限大（1/0 など）を表す。

`NaN, nan`
　数値ではないもの（0/0 など；Not a Number）を表す。

`pi`
　円周率 π を表す。

`e`

自然対数の底 e を表す。

2.15 線形代数

2.15.1 基礎的な行列演算

aa = **balance** (*a*, *opt*†)
[*dd*, *aa*] = **balance** (*a*, *opt*†)
[*cc*, *dd*, *aa*, *bb*] = **balance** (*a*, *b*, *opt*†)
　正方行列 *a* のバランス化された行列を返す。返り値が 2 つのとき `aa=aa\a*dd`，返り値が 4 つのとき `aa=cc*a*dd` と `bb=cc*b*dd` を返す。*opt* に指定できるオプションは，次の通りである。

`'N', 'n'`	バランス化しない
`'P', 'p'`	固有値を分離するために引数の順序を変える
`'S', 's'`	固有値の計算制度を向上するためにスケール化する
`'B', 'b'`	上のふたつのオプションを利用する

cond (*a*)
　行列 *a* の（2 ノルムの）条件数を返す。

det (*a*)
　行列 *a* の行列式を返す。

dmult (*a*, *b*)
　ベクトル *a* が行列 *b* の行サイズと同じとき，`diag(a)*b` を計算する。

dot (*x*, *y*)
　ベクトル *x* と *y* の内積を計算する。

lambda = **eig** (*a*)
[*v*, *lambda*] = **eig** (*a*)
　行列 *a* の固有値の集合 *lambda* と固有ベクトルの集合 *v* を計算する。

g = **givens** (*x*, *y*)
[*c*, *s*] = **givens** (*x*, *y*)
　2 × 2 の直交行列 *g* を返す。この行列は，次の構造をもつ。

$$G = \begin{bmatrix} c & s \\ -s' & c \end{bmatrix} \quad \text{であり} \quad G \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$

inv (*a*)
inverse (*a*)
　行列 *a* の逆行列を計算する。

norm (*a*, *p*†)
a の *p* ノルムを計算する。*a* が行列のとき *p* は次のようになる。

`p=1`	1 ノルム，最大の列和
`p=2`	最大特異値（デフォルト）
`p=Inf`	無限大ノルム，最大の行和
`p='fro'`	Frobenius ノルム，最大の行和

a が行列のとき *p* は次のようになる。

`p=2`	最大特異値（デフォルト）
`p=Inf`	`max(abs(a))`
`p=-Inf`	`min(abs(a))`
その他	*a* の *p* ノルム `(sum(abs(a).^p))^(1/p)`

null (*a*, *tol*†)
a の null 空間の直交基底を返す。null 空間の次元は *tol* より大きくない正則値とする。

orth (*a*, *tol*†)
a の範囲の直交基底を返す。範囲空間の次元は *tol* より大きい正則値とする。

pinv (*x*, *tol*†)
x の疑似逆行列（一般化逆行列）を返す。*tol* より小さな正則値を無視する。

rank (*a*, *tol*†)
a の階数を返す。*tol* より大きい正則値とする。

trace (*a*)
行列 *a* の対角和を計算する。

2.15.2 行列の分解

chol (*a*)
行列 *a* を Cholesky 分解（$\mathbf{A} = \mathbf{R}'\mathbf{R}$）する。

h = **hess** (*a*)
[*p*, *h*] = **hess** (*a*)
行列 *a* を Hessenberg 分解（$\mathbf{A} = \mathbf{PHP}'$）する。

[*l*, *u*, *p*] = **lu** (*a*)
行列 *a* を LU 分解する。*p* は置換行列である。

[*q*, *r*, *p*] = **qr** (*a*)

行列 *a* を QR（直交三角）分解する。*p* は置換行列である。

lambda = **qz** (*a*, *b*)
[*aa*, *bb*, *q*, *z*, *v*, *w*, *lambda*] = **qz** (*a*, *b*)
[*aa*, *bb*, *z*, *lambda*] = **qz** (*a*, *b*)
行列 *a* と *b* を，一般化固有値問題の QZ 分解する。

[*aa*, *bb*, *q*, *z*] = **qzhess** (*a*, *b*)
行列 *a* と *b* を，Hessenberg 三角分解する。

s = **schur** (*a*)
[*u*, *s*] = **schur** (*a*, *opt*†)
行列 *a* を Schur 分解する。*u* は unitary 行列である。*opt* は次の通り。

```
'a'
'd'
'u'
```

s = **svd** (*a*)
[*u*, *s*, *v*] = **svd** (*a*)
行列 *a* を特異値分解する。

[*housv*, *beta*, *zer*] = **housh** (*x*, *j*, *z*)
Householder 鏡映変換を行う。

[*u*, *h*, *nu*] = **krylov** (*a*, *v*, *k*, *epsl*†, *pflg*†)
Krylov 部分空間の直行基底を返す。

2.15.3　行列関数

expm (*a*)
行列 *a* のテイラー展開で定義される exponental を計算する。

logm (*a*)
正方行列 *a* の行列対数を計算する。

sqrtm (*a*)
正方行列 *a* の行列平方根を計算する。

kron (*a*, *b*)
行列 *a* と *b* の Kronecker 積を計算する。

x = **syl** (*a*, *b*, *c*)
Sylvester 方程式を解く。

2.16 統計

2.16.1 基礎統計量

統計関数の引数 x には，（大部分が）ベクトルと行列の両方を指定できる。

- 引数がベクトルの場合は，その各要素について該当の処理を行う。
- 引数が行列の場合は，各々の列の各要素について該当の処理を行う。

mean $(x,\ opt^{\dagger})$
x の平均値を返す。行列の場合は，列単位で計算する。opt は次の通り。

`'a'`	通常の算術平均（デフォルト）
`'g'`	幾何平均
`'h'`	調和平均

median (x)
x の中央値を返す。

std (x)
x の標準偏差を返す。

cov $(x,\ y^{\dagger})$
x と y の共分散を返す。

corrcoef $(x,\ y^{\dagger})$
x と y の相関係数を返す。

kurtosis (x)
x の尖度を返す。

mahalanobis $(x,\ y)$
x と y のマハラノビスの距離を計算する。

skewness (x)
x の歪度を返す。

values (x)
ベクトル x の重複しない値を返す。

var (x)
x の分散を返す。

$[t,\ l\_x] = \mathbf{table}\ (x)$

[*t*, *l_x*, *l_y*] = **table** (*x*, *y*)
　ベクトル *x* と *y* から，水準 *l_x*, *l_y* の分割表 *t* を作成する。

statistics (*x*)
　x の各列について，最小値，第 1 四分位数，中央値，第 3 四分位数，最大値，平均値，標準偏差，歪度，尖度を求める。

spearman (*x*, *y*†)
　x と *y* の Spearman の順位相関を返す。

run_count (*x*, *n*)
　x の要素が *n* 個以上連続して増加している個数を返す。

ranks (*x*)
　x の各要素の対応する順位を返す。

range (*x*)
　x の最小値と最大値の範囲を返す。

[*q*, *s*] = **qqplot** (*x*, *dist*, *params*)
　パラメータ *param* である分布 *dist* について標本 *x* の QQ プロット（quantile plot）を実行する。

probit (*p*)
　p の各要素についてプロビットを返す。

[*p*, *y*] = **ppplot** (*x*, *dist*, *params*)
　パラメータ *param* である分布 *dist* について標本 *x* の PP プロット（probability plot）を実行する。

moment (*x*, *p*, *opt*†)
　x の第 *p* モーメントを返す。*opt* は `'c'` が central，`'a'` が absolute を意味する。

meansq (*x*)
　x の平均平方和を返す。

logit (*p*)
　p の各要素についてロジットを返す。

kendall (*x*, *y*†)
　x と *y* から Kendall の τ を計算する。

iqr (*x*)
　x の四分位範囲を返す。

cut (*x, breaks*)
x の要素を *break* 段階のカテゴリカルデータに変換する。

cor (*x, y*†)
x と *y* の相関係数を返す。

cloglog (*x*)
x の complementary log-log 関数を返す。

center (*x*)
x を中心化（各要素から平均値を減じる）する。

2.16.2 検定

[*pval, f, df_b, df_w*] = **anova** (*y, g*†)
データベクトル *y* と分類ベクトル *g* から，1 元配置の分散分析を行う。*y* が行列で *g* を省略したとき，各列を分類とみなす（釣り合い型 ANOVA）。各分類間で平均値に差がないという帰無仮説を立てる。自由度 *df_b* と *df_w* の F-値 *f* を求め，p-値 *p* を返す。返り値の変数を設定しないとき，分散分析表を表示する。

[*pval, chisq, df*] = **bartlett_test** (*x1, ...*)
データベクトル *x1, ...* について分散等質性の Bartlett 検定を行う。分散が等しくないという帰無仮説を立てる。自由度 *df* のカイ 2 乗値 *chisq* を求め，p-値 *p* を返す。返り値の変数を設定しないとき，p-値を表示する。

[*pval, chisq, df*] = **chisquare_test_homogeneity** (*x, y, c*)
データベクトル *x* と *y* が同じ母集団からの標本である，という帰無仮説を立てて，カイ 2 乗検定を行う。エントリ *c* から導かれる分割に基づく。自由度 *df*=`length(c)` のカイ 2 乗値を求め，p-値 *p* を返す。返り値の変数を設定しないとき，p-値を表示する。

[*pval, chisq, df*] = **chisquare_test_independence** (*x*)
分割表 *x* に基づき，独立性のカイ 2 乗検定を行う。独立であるという帰無仮説を立て，自由度 *df* のカイ 2 乗値を求め，p-値 *p* を返す。返り値の変数を設定しないとき，p-値を表示する。

cor_test (*x, y, alt*†, *method*†)
データベクトル *x* と *y* が相関のない母集団からのものかどうかを検定する。*alt* には対立仮説を指定する：`'!='`, `'<>'`（ゼロではない），`'>'`（0 より大きい），`'<'`（0 より小さい）。標準では両側検定を行う。*method* に指定する検定方法は，以下の通り。

'pearson'	Pearson の積率相関係数（デフォルト）
'kendall'	Kandall の順位相関 τ
'spearman'	Spearman の順位相関 ρ

結果は構造体で返す。返り値の変数を設定しないとき，p-値を表示する。

pval	検定の p-値
stat	検定統計量の値
dist	検定統計量の分布
params	検定統計量の分布のパラメータ
alternative	対立仮説
method	検定に用いた方法

[*pval, f, df_num, df_den*] = **f_test_regression** (*y, x, rr, r*)

古典的な回帰モデル $\mathbf{y} = \mathbf{Xb} + \mathbf{e}$ について，帰無仮説 $rr\,\mathbf{b} = r$ に対する F-検定を行う。自由度 *df_num* と *df_den* の F-分布から F-値 *f* を求め，p-値 *p* を返す。返り値の変数を設定しないとき，p-値を表示する。

[*pval, tsq*] = **hotelling_test** (*x, m*)

未知の平均と分散である多変量正規分布からのデータ *x* について，平均が *m* であるかどうかの検定を行う。Hotelling の T^2*tsq* を求め，p-値 *p* を返す。返り値の変数を設定しないとき，p-値を表示する。

[*pval, tsq*] = **hotelling_test_2** (*x, y*)

分散が等しく平均が未知である多変量正規分布からの標本 *x* と *y* について，両者の平均が等しいかどうかの検定を行う。Hotelling の 2 標本 T^2*tsq* を求め，p-値 *p* を返す。返り値の変数を設定しないとき，p-値を表示する。

[*pval, ks*] = **kolmogorov_smirnov_test** (*x, dist, params, alt*†)

データベクトル *x* が連続分布 *dist* からのものであるという帰無仮説を立て，Kolmogorov-Smirnov 検定を行う。*dist* は文字列で指定する。*alt* には対立仮説を指定する：'!=', '<>'（等しくない），'>'（より大きい），'<'（より小さい）。標準では両側検定を行う。検定統計量 *ks* を求め，p-値 *p* を返す。返り値の変数を設定しないとき，p-値を表示する。

[*pval, ks, d*] = **kolmogorov_smirnov_test** (*x, y, alt*†)

データベクトル *x* と *y* が同じ連続分布からのものであるという帰無仮説を立て，Kolmogorov-Smirnov 検定を行う。*alt* には対立仮説を指定する：'!=', '<>'（等しくない），'>'（より大きい），'<'（より小さい）。標準では両側検定を行う。検定統計量 *ks* を求め，p-値 *p* を返す。*d* は 2 つの累積分布関数間の垂直距離である。返り値の変数を設定しないとき，p-値を表示する。

[*pval, k, df*] = **kruskal_wallis_test** (*x1, ...*)

1 要因の Kruskal-Wallis 分散分析を行う。

manova (*y, g*†)
多変量分散分析（MANOVA）を行う。

[*pval, chisq, df*] = **mcnemar_test** (*x*)
平方分割表データ *x* に対して McNemar の検定を行う。

[*pval, z*] = **prop_test_2** (*x1, n1, x2, n2, alt*†)
1 回めの成功数 *x1* と試行回数 *n1*，2 回めの同様の値 *x1, n2* について，成功率が等しいかどうかの検定を行う。

[*pval, chisq*] = **run_test** (*x*)
データ *x* の連続増加に基づいて自由度 6 のカイ 2 乗検定を行う。

[*pval, b, n*] = **sign_test** (*x, y, alt*†)
2 つの対応するデータ *x* と *y* について，符号検定を行う。

[*pval, t, df*] = **t_test** (*x, m, alt*†)
データ *x* について，平均が *m* である帰無仮説に対して t-検定を行う。

[*pval, t, df*] = **t_test_2** (*x, y, alt*†)
分散が等しく平均が未知である正規分布からの標本 *x* と *y* について，t-検定を行う。

[*pval, t, df*] = **t_test_regression** (*y, x, rr, r, alt*†)
古典的な回帰モデル $\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$ について，帰無仮説 $rr\,\mathbf{b} = r$ に対する t-検定を行う。

[*pval, z*] = **u_test** (*x, y, alt*†)
データ *x* と *y* について Mann-Whitney の U-検定を行う。

[*pval, f, df_num, df_den*] = **var_test** (*x, y, alt*†)
平均と分散が未知である正規分布からの標本 *x* と *y* について，等分散性の F-検定を行う。

[*pval, t, df*] = **welch_test** (*x, y, alt*†)
平均と分散が未知である正規分布からの標本 *x* と *y* について，Welch の検定を行う。

[*pval, z*] = **wilcoxon_test** (*x, y, alt*†)
2 つの対応するデータ *x* と *y* について，符号順位検定を行う。

[*pval, z*] = **z_test** (*x, m, v, alt*†)
分散が *v* で平均が未知である正規分布からの標本 *x* について，平均が *m* であるという Z-検定を行う。

[*pval*, *z*] = **z_test_2** (*x*, *y*, *v_x*, *v_y*, *alt*†)

それぞれ分散が *v_x* および *v_y* で平均が未知である正規分布からの標本 *x* と *y* について，Z-検定を行う。

2.16.3 モデル

[*theta*, *bets*, *dev*, *dl*, *d2l*, *p*] = **logistic_regression** (*y*, *x*, *print*, *theta*, *beta*)

ロジスティック回帰分析を行う。

2.16.4 分布

Octave 分布関数	分布名	引数	rnd の引数
beta	Beta 分布	*x*, *a*, *b*	*a*, *b*, *r*, *c*
binominal	二項分布	*x*, *n*, *p*	*n*, *p*, *r*, *c*
cauchy	cauchy 分布	*x*, *l*, *s*	*l*, *s*, *r*, *c*
chisquare	カイ 2 乗分布	*x*, *n*	*n*, *r*, *c*
discrete	離散分布	*x*, *v*, *p*	*n*, *v*, *p*
empirical	経験分布	*x*, *data*	*n*, *data*
exponental	指数分布	*x*, *lambda*	*lambda*, *r*, *c*
f	F-分布	*x*, *m*, *n*	*m*, *n*, *r*, *c*
gamma	Gamma 分布	*x*, *a*, *b*	*a*, *b*, *r*, *c*
geometric	幾何分布	*x*, *p*	*p*, *r*, *c*
hypergeometric	超幾何分布	*x*, *m*, *t*, *n*	*n_size*, *m*, *t*, *n*
kolmogorov_smirnov	—	*x*, *tol*	（cdf のみ）
laplace	Laplace 分布	*x*	*r*, *c*
logistic	ロジスティック分布	*x*	*r*, *c*
lognormal	対数正規分布	*x*, *a*, *v*	*a*, *v*, *r*, *c*
normal	正規分布	*x*, *m*, *v*	*m*, *v*, *r*, *c*
pascal	Pascal 分布	*x*, *n*, *p*	*n*, *p*, *r*, *c*
poisson	Poisson 分布	*x*, *lambda*	*lambda*, *r*, *c*
stdnormal	標準正規分布	*x*	*r*, *c*
t	t-分布	*x*, *n*	*n*, *r*, *c*
uniform	一様分布	*x*, *a*, *b*	*a*, *b*, *r*, *c*
weibull	Weibull 分布	*x*, *a*, *s*	*a*, *s*, *r*, *c*
wiener	Wiener 過程	（rnd のみ）	*t*, *d*, *n*

2.17 システム

2.17.1 日付と時刻

time ()
現在時刻を 1970 年 1 月 1 日 0 時 0 分（epoch）からの累積秒で返す。

ctime (*t*)
epoch からの累積秒 *t*（現地時間）を時刻文字列に変換する。

gmtime (*t*)
epoch からの累積秒 *t*（グリニッジ時刻）を構造体で返す。

localtime (*t*)
epoch からの累積秒 *t*（現地時間）を構造体で返す。

mktime (*tm_struct*)
時刻構造体を epoch からの累積秒に変換する。

asctime (*tm_struct*)
時刻構造体を時刻文字列に変換する。

strftime (*fmt*, *tm_struct*)
時刻構造体を *fmt* に従って文字列に展開する。*fmt* は以下の通り。

¥%	% 文字それ自身	%a	現地の週名（Sun-Sat）
¥n	改行文字	%A	現地の週名（Sunday-Saturday）
¥t	タブ文字	%b	現地の月名（Jan-Dec）
-	フィールドを埋めない	%B	現地の月名（January-December）
_	空白でフィールドを埋める	%c	現地の日時文字列
%H	時（00 から 23）	%C	世紀（00 から 99）
%I	時（01 から 12）	%d	月の日数（01 から 31）
%k	時（0 から 23）	%e	月の日数（1 から 31）
%l	時（1 から 12）	%D	日付（mm/dd/yy）
%M	分（00 から 59）	%h	%b と同じ
%p	AM あるいは PM	%j	1 月 1 日からの日数（001 から 366）
%r	12 時間（hh:mm:ss [AP]M）	%m	月（01 から 12）
%R	24 時間（hh:mm）	%U	1 月 1 日からの日曜の数（00 から 53）
%s	1970 年 1 月 1 日からの累積秒	%w	日曜日からの日数（00 から 06）
%S	秒；（00 から 59）	%W	1 月 1 日からの月曜の数（00 から 53）
%T	24 時間（hh:mm:ss）	%x	現地時間（mm/dd/yy）
%X	現地時間；%H:%M:%S	%y	年の下 2 桁（00 から 99）
%Z	タイムゾーン	%Y	年（1970 から）

clock ()

現在の年月日時分秒（24 時間表示）を要素とするベクトルを返す。

date ()
現在の日付を `DD-MMM-YY` の文字列で返す。

etime (*t1*, *t2*)
`clock` から返される 2 つの時刻間の差を秒で返す。

[*total*, *user*, *system*] = **cputime** ()
Octave のユーザ実行時間 *user* とシステム実行時間 *system*、その合計時間 *total* を返す。

is_leap_year (*year*)
year がうるう年ならば 1 を返す。*year* を省略すると現在の年とする。

tic ()
toc ()
最後に `tic` を実行してからの経過秒を `toc` で得る。

pause (*seconds*†)
seconds 秒だけ処理を中止する（キー入力は受け付ける）。*seconds* を省略すると、キーが押されるまで処理を中止する。

sleep ()

sleep (*seconds*†)
seconds 秒だけ処理を中止する（キー入力は受け付ける）。

usleep (*microseconds*)
microseconds$\times\frac{1}{1000}$ 秒だけ処理を中止する（キー入力は受け付ける）。

2.17.2 ファイルシステム

以下の関数は、エラー発生時に *err* に 0 以外の値、*msg* にエラー文字列が返る。

[*err*, *msg*] = **rename** (*old*, *new*)
ファイル名を *old* から *new* に変更する。

[*err*, *msg*] = **unlink** (*file*)
ファイル *file* を削除する。

[*files*, *err*, *msg*] = **readdir** (*dir*)
ディレクトリ *dir* に存在するファイル名を文字列配列 *files* で返す。

[*err, msg*] = **mkdir** (*dir*)
ディレクトリ *dir* を作成する。

[*err, msg*] = **rmdir** (*dir*)
ディレクトリ *dir* を削除する。

mkfifo (*name*)
FIFO 特殊ファイル *name* を作成する。

[*info, err, msg*] = **stat** (*file*)

[*info, err, msg*] = **lstat** (*file*)
ファイル *file* についての情報を以下の構造体 *info* で返す。

`dev`	ディレクトリエントリのデバイス ID
`ino`	ファイル番号
`modestr`	モードの文字列（`ls -l` の出力と同様）
`nlink`	リンクの数
`uid`	所有者のユーザ ID
`gid`	グループ ID
`rdev`	特殊ファイルのデバイス ID
`size`	バイト単位のサイズ
`atime`	最終アクセス日時（`time` 関数の出力形式）
`mtime`	最終更新日時（`time` 関数の出力形式）
`ctime`	最終ステータス変更日時（`time` 関数の出力形式）
`blksize`	割り当てられたブロックサイズ
`blocks`	割り当てられたブロック数

file がシンボリックリンクのときは、実際のファイルについて情報を返す。シンボリックリンク自体の情報を得るときは `lstat` 関数を用いる。

glob (*pattern*)
ワイルドカード *pattern* にマッチするファイル名を返す。

fnmatch (*pattern, string*)
文字列配列 *string* に *pattern* がマッチした要素に 1 を返す。

file_in_path (*path, file*)
パス *path* 内にファイル *file* を発見したとき、ファイルのフルパスを返す。*path* はコロン区切りの文字列とする。

tilde_expand (*string*)
文字列 *string* 内のチルダ（`~`）を展開する。

2.17.3 子プロセスの制御

[*output*, *code*] = **system** (*string*, *return_output*†, *type*†)
　文字列 *string* をシェルコマンドとして実行し、その出力文字列 *output* と終了コード *code* を返す。*type* に'async' を指定すると、プロセスがバックグラウンドで実行される。'sync'（デフォルト）を指定すると、プロセスの終了まで待つ。*return_output* が指定されるとき、コマンドからの出力が返される。

fid = **popen** (*command*, *mode*)
　コマンド *command* によりプロセスを開始してパイプを作成する。入出力ストリーム ID の *fid* を返す。*mode* の指定は以下のとおり。

'r'	パイプを標準出力に接続し、読み込みに対して開く
'w'	パイプを標準入力に接続し、書き出しに対して開く

pclose (*fid*)
　popen によって開いたパイプを閉じる。

[*in*, *out*, *pid*] = **popen2** (*command*, *args*)
　双方向のサブプロセスを開始する。*command* をコマンドとし引数を *args* で与える。入力の ID に *in* を、出力 ID に *out* を割り当て、小プロセスの ID を *pid* とする。

[*pid*, *msg*] = **fork** ()
　現在のプロセスの複製を作成する。戻り値は、以下のようになる。

> 0	現在親プロセスにおり、戻り値は子プロセスの ID である
0	現在子プロセスにおり、別プロセス実行のために exec 関数を実行できる
< 0	実行に失敗した

[*err*, *msg*] = **exec** (*file*, *args*)
　現在のプロセスを別のプロセスで置き換える。プログラム *file* の引数を *args* で与える。エラー発生時、*err* に 0 以外の値、*msg* にメッセージ文字列を返す。

[*file_ids*, *err*, *msg*] = **pipe** ()
　パイプを作成し、読み込みと書き出しに対応する ID を返す。

[*fid*, *msg*] = **dup2** (*old*, *new*)
　ファイル記述子の複製を作成し、0 以上の *fid* を返す。

[*pid*, *msg*] = **waitpid** (*pid*, *options*)

終了するためにプロセス *pid* を待機する。指定できる *pid* は以下の通り。

`-1` いずれかの子プロセスを待機する
`0` プロセスグループ ID が Octave のプロセスと等しい子プロセスを待機する
`>0` この ID をもつ子プロセスを待機する

指定できる *options* は以下の通り。

`0` シグナルを受け取るか子プロセスが終了するまで待機（デフォルト）
`1` ステータスがすぐにわからないときは終了しない
`2` 停止したプロセスのステータスを報告する
`3` 1 と 2 の両方を指定する

[*err*, *msg*] = **fcntl** (*fid*, *request*, *arg*)

開いたファイル *fid* の特性を変更する。指定できる *request* は以下の通り。

`F_DUPFD` 複製したファイル識別子を返す
`F_GETFD` ファイル識別子フラグを返す
`F_SETFD` ファイル識別子フラグを設定する
`F_GETFL` ファイルステータスフラグを返す
`F_SETFL` *arg* で指定したファイルステータスフラグを設定する

`F_GETFL` と `F_SETFL` で使用するフラグは次の通り。

`O_RDONLY` 読み込み専用で開く
`O_WRONLY` 書き出し専用で開く
`O_RDWR` 読み書き両用で開く
`O_APPEND` 追記する
`O_NONBLOCK` Nonblocking モード
`O_SYNC` 書き出しが完了するまで待つ
`O_ASYNC` 非同期 I/O

2.17.4 プロセス・グループ・ユーザ ID

getpgrp () 現在のプロセスのプロセスグループ ID を返す。
getpid () 現在のプロセスのプロセス ID を返す。
getppid () 親プロセスのプロセス ID を返す。
geteuid () 現在のプロセスの実行ユーザ ID を返す。
getuid () 現在のプロセスの真のユーザ ID を返す。
getegid () 現在のプロセスの実行グループ ID を返す。
getgid () 現在のプロセスの真のグループ ID を返す。

2.17.5　環境変数

getenv (*var*)
　環境変数 *var* の値を返す。

putenv (*var, value*)
　環境変数 *var* に値 *value* を設定する。

2.17.6　現在のディレクトリ

cd *dir*
chdir *dir*
　現在の作業ディレクトリを *dir* に移動する。

ls *options*
dir *options*
　現在のディレクトリ内容を表示する。

pwd ()
　現在の作業ディレクトリ名を返す。

2.17.7　パスワードデータベース

[*pw_struct*] = **getpwent** ()
　パスワードデータベースからエントリを含む構造体を読み出す。

[*pw_struct*] = **getpwuid** (*uid*)
　ユーザ ID*uid* のパスワードデータベースからの最初のエントリを含む構造体を読み出す。

[*pw_struct*] = **getpwnam** (*name*)
　ユーザ名 *name* のパスワードデータベースからの最初のエントリを含む構造体を読み出す。

setpwent ()
　パスワードデータベースの始点への内部ポインタを返す。

endpwent ()
　パスワードデータベースを閉じる。

2.17.8 グループデータベース

[*pw_struct*] = **getpwent** ()
グループデータベースからエントリを含む構造体を読み出す。

[*pw_struct*] = **getpwuid** (*gid*)
グループ ID*gid* のグループデータベースからの最初のエントリを含む構造体を読み出す。

[*pw_struct*] = **getpwnam** (*name*)
グループ名 *name* のグループデータベースからの最初のエントリを含む構造体を読み出す。

setpwent ()
グループデータベースの始点への内部ポインタを返す。

endpwent ()
グループデータベースを閉じる。

2.17.9 システム情報

computer ()
CPU-Vendor-OS の情報を文字列で返す。

isieee ()
浮動小数点演算が IEEE に基づくコンピュータであれば 1 を返す。

octave_config_info (*option*†)
Octave の設定とインストール情報の構造体を返す。*option* を指定すると特定の項目のみ返す。

getrusage ()
現在の Octave プロセス情報の構造体を返す。

2.18 組み込み変数

2.18.1 起動・終了・全般

argv
　Octave の起動オプション文字列

program_invocation_name
program_name
　実行しているプログラム名

INFO_FILE
　Octave info file の位置

INFO_PROGRAM
　実行する info program の名前

MAKE_INFO_PROGRAM
　Texinfo でマークアップされたヘルプを実行するプログラム名

supress_verbose_help_message
　この値が 0 でないとき，help による表示に追加内容を表示しない

completion_append_char
　コマンドの補完後に追加入力される文字列

EDITOR
　編集に使用するエディタのプログラム名

history_file
　コマンド履歴を保存するファイル名

history_size
　保存する履歴の最大数

saving_history
　この値が 0 でないとき，履歴をファイルに保存する

PS1
　通常コマンド入力時のプロンプト表示形式

PS2
　追加入力が必要な場合のプロンプト表示形式

PS4

--echo-input 指定時，入力前に表示されるプロンプト表示形式

2.18.2 データのサイズと型

whitespace_in_literal_matrix
リテラル行列の表記方法の指定

warn_separator_insert
リテラル行列に自動的にカンマとセミコロンを挿入するときに警告する

output_max_field_width
数値の出力幅

output_precision
小数の出力精度

split_long_rows
この値が 0 でないとき，行列の出力を自動的に分割する

fixed_point_format
この値が 0 でないとき，すべての数値を同時に同じ尺度で表示する

print_empty_dimension
この値が 0 でないとき，空行列の次数を表示する

empty_list_elements_ok
この値が 0 でないとき，空行列を無視する

propagate_empty_matrices
この値が 0 でないとき，inverse のような関数に影響する

true
真の論理値

false
偽の論理値

string_fill_char
文字列行列の要素の長さをそろえるために挿入する文字

2.18.3 文字列

implicit_num_to_str_ok
この値が 0 でないとき，数値から ASCII 文字への暗黙の変換を行う

implicit_str_to_num_ok
　この値が 0 でないとき，ASCII 文字から数値への暗黙の変換を行う

warn_single_quote_string
　文字定数にシングルクオートを用いたときに警告する

2.18.4 構造体

struct_levels_to_print
　表示する構造体のレベル

2.18.5 変数

initialize_global_variables
　この値が 0 でないとき，グローバル変数には初期値が設定される

default_global_variable_value
　グローバル変数の初期値

2.18.6 表現

do_fortran_indexing
　この値が 0 でないとき，2 次元行列の添え字を 1 次元で指定する

prefer_zero_one_indexing
　この値が 0 でないとき，添え字の指定を 0 か 1 かで表す

prefer_column_vector
　この値が 0 でないとき，明示しないベクトルは列ベクトルになる。

resize_on_range_error
　この値が 0 でないとき，変数サイズを自動的に大きくする

max_recursion_depth
　再帰の深さの最大値

warn_divide_by_zero
　この値が 0 でないとき，0 で除算したときに警告が発生する

print_rhs_assign_val
　この値が 0 でないとき，左辺の代わりに右辺の値を表示する

2.18.7 評価

default_eval_print_flag
この値が 0 でないとき，**eval** の結果を表示しない

2.18.8 ステートメント

warn_assign_as_truth_value
この値が 0 でないとき，if(s = t) のような記述に警告する

warn_variable_switch_label
この値が 0 でないとき，switch のラベルが定数でないときに警告する

2.18.9 関数とスクリプトファイル

silent_functions
この値が 0 でないとき，関数からの内部表示を抑制する

warn_missing_semicolon
この値が 0 でないとき，関数定義内の文がセミコロンで終わっていないときに警告する

default_return_value
標準で関数が返す値

default_all_return_value
この値が 0 でないとき, 関数が返すすべての値を default_return_value とする。

return_last_computed_value
この値が true ならば，関数内で最後に表現した値を返す

DEFAULT_LOADPATH
デフォルトのファイル検索パス（コロン区切り）

LOADPATH
ファイル検索パス（コロン区切り）

ignore_function_time_stamp
関数ファイルを再コンパイルする基準を設定

warn_function_name_clash
　この値が 0 でないとき，関数ファイル名と関数名が一致していないときに警告する

warn_future_time_stamp
　この値が 0 でないとき，関数ファイルの日付が未来であれば警告する

warn_reload_forces_clear
　同一ファイルから複数の関数を強制的に再読み出しする

variables_can_hide_functions
　この値が 0 でないとき，変数が以前に呼び出した同名の関数を上書きする

2.18.10 エラー処理

error_text
　try などで発生したエラー内容を含む

beep_on_error
　この値が 0 でないとき，エラー発生時にビープ音を鳴らす

2.18.11 入出力

PAGER
　ページ単位に表示するためのプログラム（ページャ）名

page_screen_output
　この値が 0 でないとき，画面に入りきらない出力をページャに送る

page_output_immediately
　この値が 0 でないとき，出力をすぐにページャに送る

print_answer_id_name
　この値が 0 でないとき，結果とともに変数名を表示する。

crash_dumps_octave_core
　この値が 0 でないとき，クラッシュ時に全変数をファイルに出力

default_save_format
　save コマンドで保存する標準の形式

save_precision
　保存時の数値精度

2.18.12 プロット

automatic_replot
: この値が 0 でないとき，プロットすべき数値が更新されると再描画する

gnuplot_binary
: gnuplot のプログラム名

gnuplot_has_frames
: この値が 0 でないとき，gnuplot が複数ウインドウ対応しているとする

gnuplot_has_multiplot
: この値が 0 でないとき，gnuplot が多重プロット対応しているとする

2.18.13 行列演算

treat_neg_dim_as_zero
: この値が 0 でないとき，負の添え字をゼロと見なす（空行列とする）。

ok_to_lose_imaginary_part
: この値が 0 でないとき，暗黙のうちに複素数を実数に変換する。

2.18.14 算術演算

eps
: 計算機の精度

realmax
: 実数の最大値

realmin
: 実数の最小値

2.18.15 システム

EXEC_PATH
: 子プロセスの実行可能なパス（コロン区切り）

OCTAVE_VERSION
: Octave のバージョン

索引