> **Bash基礎文法最速マスター**
> Bash

Bashの文法一覧です。他の言語をある程度知っている人はこれを読めばBashの基礎をマスターしてBashを書くことができるようになっています。簡易リファレンスとしても利用できると思いますので、これは足りないと思うものがあれば教えてください。

1.基礎

echo文

echo文です。

```
echo "Hello world"
```

コメント

コメントです。

```
# コメント
```

変数の宣言

変数の宣言です。

```
# 文字列変数
declare str

# 数値変数
declare -i num

# 配列変数
declare -a array
```

スクリプトの実行

スクリプトを実行するにはコマンドラインで次のようにします。

```
bash script.sh
```

出力結果をファイルに書き出すにはリダイレクトを使います。

```
bash script.sh > file.txt
```

文法チェック

事前に文法チェックを行うには、次のようにします。

```
bash -n script.sh
```

デバッグ

デバッグ用に実行中のコマンドを表示するには、次のようにします。

```
bash -x script.sh
```

2.数値

数値の表現

変数に数値を代入できます。変数には整数のみ代入できます。

```
declare -i num

num=2
num=100000000
```

四則演算

四則演算です。

```
num=1+1
num=1-1
num=1*2
num=1/2
```

余りの求め方です。

```
num=1%2
```

インクリメントとデクリメント

インクリメントとデクリメントです。

```
# インクリメント
let ++num

# デクリメント
let --num
```

3.文字列

文字列の表現

文字列はそのまま書くか、シングルクォートかダブルクォートで囲みます。文字列にスペース文字やタブ文字などの単語区切り文字が含まれる場合は、シングルクォートかダブルクォートで囲む必要があります。

文字列をそのまま書いた場合とダブルクォートで囲んだ場合は、変数展開が行われます。文字列をシングルクォートで囲んだ場合は変数展開は行われません。

```
declare str1=abc        # abc
declare str2="de f"     # de f
declare str3='g hi'     # g hi

declare str4=$str1      # abc
declare str5="$str1"    # abc
DECLARE str6='$str1'    # $str1
```

文字列操作

各種文字列操作です。

```
# 文字列連結
declare str_a='aaa'
declare str_b='bbb'
declare join1=$str_a$str_b

declare -a array=(aaa bbb ccc)
OLDIFS=$IFS
IFS='|'
declare join2="${array[*]}"
IFS=$OLDIFS

# 文字列分割
OLDIFS=IFS
IFS=','
declare str='aaa,bbb,ccc'
declare -a record=($str)
IFS=OLDIFS

# 長さ
declare str=abcdef
declare -i length=${#str}

# 切り出し
declare str=abcd
declare substr=${str:0:2}

# 検索
declare str1='abcd'
declare str2='cd'
OLDIFS=IFS
IFS="$str2"
declare temp=($str1)
declare -i str1_len=${#str1}
declare -i temp_len=${#temp}
IFS=OLDIFS
if [ $str1_len -eq $temp_len ]
then
    declare -i result=-1         # 見つからなかった場合は-1
else
    declare -i result=$temp_len  # 見つかった場合はその位置
fi
```

4.配列

配列の変数の宣言と代入

配列です。

```
# 配列の宣言
declare -a array

# 配列への代入
array=(1 2 3)
```

配列の要素の参照と代入

配列の要素の参照と代入です。

```
# 要素の参照
${array[0]}
${array[1]}

# 要素の代入
array[0]=1
array[0]=2
```

配列の要素の個数

配列の要素の個数です。

```
declare -i array_num=${#array[@]}
```

配列の操作

配列の操作です。

```
declare -a array=(1 2 3)

# 先頭を取り出す
first="${array[0]}"            # firstは1
array=("${array[@]:1}")        # arrayは(2 3)

# 先頭に追加
array=(5 "${array[@]}")        # arrayは(5 2 3)

# 末尾を取り出す
declare -i num=${#array[@]}-1
last="${array[$num]}"          # lastは3
array=("${array[@]:0:$num}")   # arrayは(5 2)

# 末尾に追加
array=("${array[@]}" 9)        # arrayは(5 2 9)
```

5. 制御文

if文

if文です。

```
if [ 条件 ]; then

fi
```

if 〜 else文

if 〜 else 文です。

```
if [ 条件 ]; then

else

fi
```

if 〜 elif文

if 〜 elif 文です。他の言語のようにelse ifではなくelifであることに注意しましょう。

```
if [ 条件 ]; then

elif [ 条件 ]; then

fi
```

while文

while文です。

```
declare -i i=0
while [ $i -lt 5 ]; do

    # 処理

    let ++i
done
```

for文

for文です。

```
declare -i i
for (( i=0; $i < 5 ; ++i )); do

done
```

for each文

for each文です。

```
for field in "${fields[@]}"; do

done
```

6.関数

関数です。

Bashでは他の言語のように明示的に引数を指定することはありません。$1、$2、$3などのパラメータに引数が格納されるので自力で取り出します。

Bashの関数では戻り値を返すことはできません。代わりに、終了ステータスを返すことができます。終了ステータスを返すにはreturnを使います。終了ステータスには0～255の整数を指定できます。

関数を作るには次のようにします。

```
function show_sum {
    declare -i num1=$1
    declare -i num2=$2

    declare -i total=num1+num2

    echo $total

    return 0
}
```

7.ファイル入出力

ファイル入出力です。

以下がファイル入力の雛形になります。

```
declare line
while read line
do

done < $file
```

以下がファイル出力の雛形になります。

```
cat /dev/null > $file

echo "aaa" >> $file
echo "bbb" >> $file
echo "ccc" >> $file
```

知っておいたほうがよい文法

Bashでよく出てくる知っておいたほうがよい文法の一覧です。

コマンドライン引数

コマンドライン引数を受けとるには$1、$2、$3などのパラメータを使用します。

```
declare file=$1
declare options=$2
```

until文

until文はwhile文の否定を表します。

```
until [ 条件 ]; do

done
```

終了ステータス

直前のコマンドや関数の終了ステータスを取得するには以下のようにします。

```
status=$?
```

IFS(内部フィールド区切り文字)

IFSは単語分割用の区切り文字を格納する変数です。Bashの組み込みコマンドが単語分割を行うときにIFSの値を使用します。デフォルトでは空白文字、タブ文字、改行文字が格納されています。

区切り文字を指定して単語分割を行いたい場合、その区切り文字を一時的にIFSに代入します。単語分割が完了したら、後続の処理に影響がないようにIFSの値を元に戻します。

```
# カンマで分割
OLDIFS=IFS
IFS=','
declare str='aaa,bbb,ccc'
declare -a record=($str)
```

Bash参考資料

- Bashで覚えておくとよいデータ構造 - 配列
- Bashを使うなら理解しておきたいアルゴリズム - 抽出・ソート・結合・集計

基礎文法最速マスターリンク集

この記事は最近の基礎文法最速マスターの流れに便乗して作成したものです。

以下、各種基礎文法最速マスターへのリンクです。

- Perl基礎文法最速マスター - Perl入門～サンプルコードによるPerl入門～
- Route 477 - Ruby基礎文法最速マスター
- PHP基礎文法最速マスター | Shin x blog
- Python基礎文法最速マスター - D++のはまり日誌
- Java基礎文法最速マスター - 何かしらの言語による記述を解析する日記
- VBA基礎文法最速マスター - 何かしらの言語による記述を解析する日記
- Brainf*ck基礎文法最速マスター - 医者を志す妻を応援する夫の日記
- Haskell基礎文法最速マスター - think and error
- Bash基礎文法最速マスター - 何かしらの言語による記述を解析する日記
- VBScript 基礎文法最速マスター - CX's VBScript Diary - VBScript グループ
- JavaScript基礎文法最速マスター - なんとなく日記