

～シェルスクリプト～

1.1 変数

◆「変数」とは文字や数字など、値を入れておく箱のようなものです。

- 箱に着けた名前を「変数名」
- 箱の中に入っているものを変数の「値」と呼びます。

環境変数

…bashが始まってから、終了するまでしか値が定義されない変数。

シェル変数

…シェルが始まってから、終了するまでしか値が定義されない変数。

★例（シェル変数の定義）

```
Num=123
```

上記の例では、変数「Num」を定義し、初期値として123を代入しています。
変数は自由な名前にしてください。

★例（環境変数の定義）

```
export KNum=234
```

環境変数の場合は、「export」コマンドを使用します。

★変数の確認の仕方

echo Num

※環境変数でも、シェル変数でもこれで確認ができる。

★今の変数が何が定義されているのかの確認

env

※一覧をまとめて確認できます。

★ここまでが、シェルを書く上で覚えておくべきコマンドと変数です。

1.2 シェル関数の定義

◆「関数」とは、何度も使う処理をひとまとめに登録しておく機能です。

```
function fNum () {定義したい内容}
```

```
function fNum () { ls -l }
```

•変数では、値しか定義できないが、
関数であればコマンドを(スペースを含めて)定義できる。

1.3 引用符とコマンド置換

◆変数を指定するときに使用する引用符によって挙動が違います。

- 「`」 シングルクォーテーション
- 「"」 ダブルクォーテーション
- 「`」 バッククォーテーション

★「 ` 」 シングルクォーテーション

◆VAR変数に、「date」コマンドを代入してみています。

```
VAR=date
```

・引用符なしの場合

```
echo $VAR  
date
```

・シングルクォーテーション

```
echo '$VAR'  
$VAR
```

…**囲った文字列をそのまま出力してくれます。**

★「”」 ダブルクオーテーション

•ダブルクオーテーション

```
echo "$VAR"
```

```
date
```

…代入した値を出力してくれます。

★「`」 バッククォーテーション

•バッククォーテーション

```
echo ` $VAR `
```

2019年 7月9日 火曜日 00:00:00 JST

…値がコマンドである場合、実行までしてくれます。

1.4 テキストファイル

ファイルは2種類存在します。

◆テキストファイル

… 人間が認識できる言語で書かれる。

◆バイナリファイル

… コンピュータが認識できるように書かれていて、人間の目には文字化けをして表示される。

※音楽ファイル、画像ファイル、動画ファイル、イメージファイルなど

シェルスクリプトは、
テキストファイルをbashが自動コンパイルしてくれるので、バイナリ変換など
が必要ありません。

2.1 シェルスクリプトの書き方

テキストファイル（メモ帳など）に以下のような形式で書いていきます。
Linux上では、「viコマンド」で記入が可能です。

`#!/bin/bash`

①bashを活用することを宣言。

`VAR=date`
`echo $VAR`

`echo $?`

②実行したい本文。
変数や関数、そのほかコマンドは自由に
設定していきます。

2.2 シェルスクリプトの実行の仕方

```
bash test.bash
```

拡張子は、基本的にbashです。
ファイル名は自由です。

```
source test.bash
```

※実行時に引数を引き渡すこともできます。

```
bash test.bash date
```

```
#!/bin/bash
```

```
VAR=$1  
echo $VAR
```

第1引数dateが引き渡されています。

```
echo $?
```

★引数に対応する変数

スクリプト内に、以下の特定の変数を入れておくと引数が代入されます。

変数名	説明
\$0	シェルスクリプトのファイル名
\$1	1番目の引数。(以降、「\$ 2」が2番目。
\$#	引数の数。

2.3 戻り値(正常確認)

◆スクリプトは、実行してみると一瞬ですべての処理を順番に実施してくれるのがメリットです。

◆一方で、コーディング中は一瞬で終わるが、うまくいっているかがわかりません。

そこで、echo \$?を最後に記載することで、画面上に正常に終わったかどうかの戻り値が返ってきます。
(1ならエラー。0なら正常。)

#!/bin/bash

VAR=\$1
echo \$VAR

echo \$?

2.4 ファイルのチェック

◆ **test 条件文** または、 **[条件文]**

条件文に記述された式を評価し、結果が真なら「0」を、偽なら「0以外」を返します。

1,2が返ってきたりしたら、それぞれLinuxの開発もとで定められているエラーコードなので、サポート問い合わせなどで調べましょう。

★条件式一覧

条件式	説明
-f	通常ファイルが存在すれば「真」
-d	ディレクトリとして存在すれば「真」
数値 1 -eq 数値2	数値1と数値2が等しければ「真」
数値 1 -ge 数値2	数値1が数値2より大きい、またはひとしければ「真」
数値 1 -ne 数値2	数値1と数値2が等しくなければ「真」
文字列1 = 文字列2	2つの文字列が等しければ「真」
文字列1 != 文字列2	2つの文字列が等しくなければ「真」

条件式	説明
条件1 -a 条件2	両方の条件式が「真」であれば「真」
条件1 -o 条件2	いずれかの条件式が「真」であれば「真」

3.1 if文

◆条件によって、処理が分岐する文章です。

※最初はこれだけできるようになれば、スクリプトは書けます。

```
if 条件式 ; then
    実行処理1
else
    実行処理2
fi
```

★if文例

```
bash test.bash 1
```

```
-----
```

```
#!/bin/bash
```

```
VAR=$1
```

```
if [ "$VAR" -eq 1 ] ; then
```

```
    echo “処理成功”
```

```
else
```

```
    echo “処理エラー”
```

```
fi
```

```
echo $?
```

3.2 case文

◆条件の分岐が3つ以上になる場合に使用します。

```
case 式 in
  パターン1)
    実行処理1;;
  パターン2)
    実行処理2;;
esac
```

★case文例

```
bash test.bash start
```

```
-----
```

```
#!/bin/bash
```

```
VAR=$1
```

```
case "$VAR" in
```

```
start)
```

```
    echo “スタートします。”::
```

```
stop)
```

```
    echo “ストップします。”::
```

```
*)
```

```
    echo “startかstopを引数に指定してください。”::
```

```
esac
```

```
echo $?
```

★Linuxの基本コマンド(スクリプトではこれくらい押さえておけば大丈夫です。)

◆現在地のファイル一覧を確認するコマンドは「**ls -l**」である。

◆/root/userに移動するコマンドは、「**cd /root/user**」である。

◆現在地を確認できるコマンドは「**pwd**」

◆ファイル名 test.bashを作成するコマンドは、「**touch test.bash**」である。

◆現在地に、ディレクトリ名 20190101を作成するコマンドは「**mkdir 20190101**」である。

◆ファイルを開いて、test.bashなどのエディタ(メモ帳など)を編集するコマンドは「**vi test.bash**」

◆ファイルAをBとしてコピーできるコマンドは「**cp A B**」である。

◆ファイルのtest.bashを開くコマンドは、「**cat test.bash**」である。

★練習

- ①基本コマンドを回答してみよう。
- ②練習問題 if文を解いてみよう。
- ③練習問題 case文を解いてみよう。
- ④自分で、書きたいスクリプトを書いてみよう。

★まとめ

◆シェルスクリプトは、基本コマンドや簡単な使い方の組み合わせです。

※なので、いろんな発想を自分で試してみてください。

◆現場の先輩などで上手な人がいれば、その人の書き方を真似してみま
しょう！！

※コーディング(プログラミング)と同じ発想なので、処理が少なくなったほう
がサーバに負荷がかからなくなるので、評価されます。

◆何回も書いて失敗して、デバッグ(修正)をしましょう。

※現場だとそんな時間がないので、自宅のPCでかけるようにして練習する
必要があります。

★勉強の方向性

◆シェルスクリプトを練習できる環境を持ってない方は、Linux環境構築を勉強しましょう。

◆シェルスクリプトを勉強する理由があいまいな方には、あまりお勧めしません。

※必要な技術がなんなのかをはっきりさせてから勉強しましょう。
(時間と費用コストが大きくなって、挫折しやすいです)

◆目の前の業務をできるようにするために学習するのも、何年も先まで必要な技術であるものなら、専門技術を学ぶことをお勧めします。