この章で学ぶこと

Shell Scriptの文法を学んで、実務でシェルを作成できるようになることが目標

・ファイルの作成、実行、コメント文
・変数の扱い
・配列の扱い
・引数の扱い
・標準入出力、ファイル入出力
・if文、and, or, ファイル存在チェック
・数値計算
・case文
・ループ文(while, until, for)
・select
・関数(function)
・local variable, read only
・PID, trap,
・debug

shell scriptとは？

Linux OSを操作するための簡易なプログラミング言語。
shell scriptの中でも、bash, sh, csh, zshがあり若干書き方が異なる

この講座では、bashを扱います。

shell scirpt 基本 その1(ファイル作成、実行、コメント)

【サポートしているシェルを一覧表示】
cat /etc/shells

【ファイル作成(viコマンドでファイル名末尾に.shを付ける)】
↓ファイルの中身
```
#! /bin/bash　**#1行目でシェルの場所を指定する**

echo 'Hello World'　**# Hello World表示**
**# コメント文は#を文頭に記載**
exit 0 **#処理終了(0が正常終了、それ以外は異常終了)**
```
【実行】
chmod 755 ファイル名**#パーミッション変更(パーミッションについての詳細はLPICの章で)**

./test.sh **#シェルスクリプトの実行**

shell scirpt 基本 その2(変数)

【スクリプト内での変数の定義】

var1=‘変数1’　**#変数の宣言(var1という名前の変数に文字列’変数1‘が格納される)**

var2=`command` **#コマンドの実行結果を変数に格納**

var3=$(command) **#コマンドの実行結果を変数に格納**

echo $var1 **# 変数の中身を取り出す場合には、$を付ける**

【システム変数の表示】

*) 初めから、定義されている変数

$BASH, $BASH_VERSION, $HOME, $PWD　**#システム変数**

shell scirpt 基本 その3(配列[Array])

【配列の作成】

fruits=(‘banana’ ‘apple’ ‘grape’)

【配列の表示方法】

echo “${fruits[@]}” **# 配列の中の値を全て表示(banana apple grape)**

echo “${fruits[0]}” **#インデックス0の要素を表示（1番目の要素(banana)）**

echo “${!fruits[@]}” **#インデックスを表示(0 1 2)**

echo “${#fruits[@]}” **#配列の要素の数を表示(3)**

fruits[3]=‘lemon’ **# 配列のインデックス3(4番目)にlemonを追加**

unset fruits[2] **# 配列のインデックス2(3番目)を削除**

shell scirpt 基本 その4(引数)

【引数とは】

実行する際に、シェルに渡す値。 ./〇〇.sh a b # このa,bがそれぞれ第1引数、第2引数

echo $1 $2 $3 **#第1引数、第2引数、第3引数を表示**

echo $0 **# シェルスクリプトのファイル名**

echo $@ **#全引数を表示**

echo $# **#引数の数を表示**

$? **#直前に実行したコマンドの戻り値(成功の場合0)**

シェルスクリプトから別のシェルスクリプトを呼び出すことも可能（やってみる）

shell scirpt 基本 その5(標準入出力、ファイル出力)

【標準入出力】
標準入力: ターミナルからの値の読み取り。標準出力: ターミナルでの値の表示
read var　**# 標準入力**
echo var1 = $var　**# 入力結果表示**

read var1 var2 var3 **# 入力結果から3個変数を設定**
echo var1 = $var1, var2 = $var2, var3 = $var3

read –p ‘var1 : ’ var1　**# 文字付きの標準入力**
read –sp ‘password : ‘ password　**# シークレットモードでの標準入力**
read –a names　**# 配列(Array)入力**

【ファイル出力】
>: 上書き
echo ‘hello’ > hello.txt **# 現在いるディレクトリのhello.txtファイルを作成(上書き)してhelloを書き込む**
>>: 追記　**# 現在いるディレクトリのhello.txtファイルを作成(追記)してhelloを書き込む**

shell scirpt 基本 その6(if文)

if文の書き方(3通り)
① if test 条件文; ② if [条件文]; ③ if [[条件文]];

if test 条件文; or [条件文]; or [[条件文]];
then
　　プログラム **# ifの条件文を満たすときに実行される**
elif test 条件文; or [条件文]; or [[条件文]]; **# ifの条件文を満たしていない場合に実行される**
then
　　プログラム **# elifの条件文を満たす場合に処理が実行される**
else
　　プログラム　**# ifもelifも満たさない場合に実行される**
fi

= != **#文字列で等しいか等しくないか**
-eq(=), -ne(!=), -lt(<), - le(≦), –gt(＞), –ge(≧) **# 数値比較**

shell scirpt 基本 その7(and or)

条件文1 and 条件文2の場合、条件文1,条件文2を両方満たす場合に実行

書き方は、以下の4通り

if [条件文1] && [条件文2];

if [条件文1 –a 条件文2];

if [[条件文1 && 条件文2]];

if test 条件文1 && test 条件文2;

条件文1 or 条件文2の場合、条件文1,条件文2のどちらかを満たす場合に実行

書き方は、以下の4通り

if [条件文1] || [条件文2];

if [条件文1 –o 条件文2];

if [[条件文1 || 条件文2]];

if test 条件文1 || test 条件文2;

否定を表す条件文(3つ)

if ! test 条件文、 if [! 条件文]、if [[! 条件文]]

cmd1 && cmd2 **#cmd1が正しく実行されたらcmd2を実行**

cmd1 || cmd2 **#cmd1がエラーならcmd2を実行**

cmd1 && cmd2 || cmd3 **#cmd1が正しく実行されたらcmd2が実行されて、cmd1がエラーならcmd3を実行**

shell scirpt 基本 その8(ファイルの存在チェック)

```
if [ -e ファイル名 ]; #ファイルもしくはディレクトリの存在確認
then
else
fi


if [ -f ファイル名] #ファイルが存在しディレクトリでなくファイル
if [ -d ファイル名 ] #ディレクトリが存在するか
if [ -s ファイル名 ] #ディレクトリ or 中身のあるファイルか
if [ -wファイル名 ] #書き込み権限あるか
if [ -xファイル名 ] #実行権限あるか
if [ fileA –nt fileB] #fileAがfileBより新しいか
if [ fileA –ot fileB] #fileAがfileBより古いか
```

shell scirpt 基本 演習問題1(exam1)

- 引数を２つ取ります
- 1つ目の引数が性別(man, woman)、2つ目の引数が年齢(age)
- 第1引数がmanの場合、Manと表示、womanの場合Womanと表示
- 第2引数が20未満の場合Child、20以上60未満の場合Adult、60以上の場合はElderlyと表示してください。
- 標準出力はコロン(:)でつないで表示してください。例えばMan:Adult, Woman:Childと表示します。
- 以下の場合はexit 1で終了してください。
  - 引数が2つでない場合
  - 1つ目の引数がman woman以外の場合
  - 2つ目の引数が0以上でない場合

./ex1.sh man 20 => Man:Adult

./ex1.sh woman 70 => Woman:Elderly

./ex1.sh man 15 => Man:Child

shell scirpt 基本 演習問題2(exam2)

- まず、現在いるディレクトリのファイル一覧を表示します
- 2回、値を標準入力で受け付けて、変数に設定します
- 1つ目の入力がファイル名とし、2つ目の入力は任意の値とします。
- ファイルが存在する場合は、そのファイルに2つ目の値を追記します。
- ファイルが存在しない場合は、「ファイルが存在しません」と表示してください。

shell scirpt 基本 その9(数値計算1)

【シェルスクリプトでの四則演算】
シェルスクリプトでは、シンプルに1+1と書いても、計算をしてくれない。四則演算をするには、少し特殊な書き方が必要

echo 1+1 **# 1+1と表示されて、計算結果(2)が表示されない**
echo $((1 + 1)) **#2と表示される。 +(和),-(差),*(積),/(商),%(剰余)**
echo $(expr 1 + 1) **#2と表示される。 +(和),-(差),¥*(積),/(商),%(剰余)**

【変数を使った四則演算】
num1=1
num2=2
echo $((num1 + num2)) **+(和),-(差),*(積),/(商),%(剰余)**

echo $(expr $num1 + $num2) **+(和),-(差),¥*(積),/(商),%(剰余)**
echo $(expr $num1 ¥* $num2)

shell scirpt 基本 その10(数値計算2)

【複雑な数値計算】

ここでは、単純な四則演算でなくより複雑な数値計算（浮動小数点演算等）を実行してみます

bcコマンドのインストール(yum install bc)

bcコマンドは、任意の精度（小数点以下の桁数を指定）の数値を扱い、四則演算・平方根・三角関数など様々な数学関数を計算できるコマンドです。

計算内容を文字列として|を使って、bcコマンドに渡すと計算されます。

echo “20.5+5” | bc # **bcコマンドで文字列20.5+5を計算する(25.5)**

echo “20.5*5” | bc # **bcコマンドで文字列20.5*5を計算する(102.5)**

echo “scale=20;sqrt(20.5/5)” | bc -l

 # -lを利用すると標準数学ライブラリを読み込んでより複雑な数学関数を実行します

echo “$num1+$num2” | bc # **変数を使用しても、bcコマンドを利用できます。**

shell scirpt 11(case)

ある変数を評価して、値に応じて処理を変える

```
case $var in # $varはチェックする変数
  パターン1 ) # $varがパターン1に該当する場合に実行される
    プログラム1 ;;
  パターン2 ) # $varがパターン2に該当する場合に実行される
    プログラム2 ;;
  * ) # $varがパターン1にもパターン2にも該当しない場合に実行される
    プログラム3 ;;
esac
```

パターンの記載方法

[a-z]: a-zまでの場合
[A-Z]: A-Zまでの場合
[0-9]:0-9までの場合

shell scirpt 12(while, break, continue)

while文はループ文の一種でwhileの条件を満たしている限り中の処理を実行し続けます

while [文字列の比較] ((数値の比較)) **# nが10以下の場合にwhileの中の処理を実行**
do
　　プログラム
　　sleep 1 **#1秒処理が止まる**
done

 # ファイルの読込み(変数pの中にファイルの中身が1行ずつ読み込まれます)
while read p
do
　　プログラム
done < **aa.txt**
cat aa.txt | while read p
do
 statement
done

shell scirpt 13(until, break, continue)

until文はループ文の一種でuntilの条件を満たすまで中の処理を実行し続けます

until [文字列の比較] ((数値の比較))

do

　　statement

done

break: while文、until文の中にbreakを入れるとそれが実行された時点でループの外に出ます

continue: while文、until文の中にcontinueを入れるとそこでループの以降の処理が実行されず、次のループに移ります

shell scirpt 14(for, break, continue)

for文はループ文の一種で条件にある値を変数格納してループ内の処理を実行し続けます

for var in 1 2 3 4 5 **変数varに1,2,3,4,5を代入していきます**

 # seq 1 10: 1~10までの整数。seq 1 2 10: 1 3 5 7 9
for i in `seq 1 10`, for i in `seq 1 2 10`

for command in ls pwd date; **#変数commandにls pwd dateを代入していきます**
do
  echo $command
  $command
done

shell scirpt 基本 演習問題3(exam3)

Fizz Buzz問題

- 1-100までループする
- 数値が3の倍数の場合は**数値: Fizz**と表示
- 数値が5の倍数の場合は**数値: Buzz**と表示
- 数値が15の倍数の場合は**数値: FizzBuzz**と表示
- 上の条件に満たさない場合は、数値のみ表示

・for while untilそれぞれ使用して作成する

shell scirpt 基本 演習問題4(exam4)

Fileの中身を計算する問題

以下のような数値を1列に並べたファイルを作成する
1
2
:
・ユーザから標準入力(ファイル名)を受け、ファイルの存在を確認する
・さらに標準入力を受け取り、その標準入力がsumの場合はファイルの合計を計算
・avgの場合は平均値を計算
・minの場合は最小値
・maxの場合は最大値を計算
・exitで処理を終了する
(ただし、ファイルの中には1~100までの数値が入っており、必ず1つ以上の数値が入っている事は前提とする)

shell scirpt 15(select)

selectを利用すると複数の文字の中から、どれを選ぶのか数値で選択することができる。ユーザに数値を選択させて、それに応じて処理をさせたい場合に用いる

select var in apple banana lemon **#ユーザが入力して、 1) apple 2) banana 3) lemonの中から選択する（変数varに1の場合はapple、2の場合はbanana、3の場合はlemon、1,2,3以外の入力の場合空白が入る）**

```
do
   echo $var
   break # breakでselectの外に出る
done
```

shell scirpt 16(function)

関数を定義(function 関数名())

```
function hello(){
        プログラム
        echo $1 # 第一引数
}



hello # 関数の呼び出し(引数なし)
hello ‘hello’ # 第一引数を’hello’で関数helloの呼び出し
```

shell scirpt 17(local variable)

```
# 関数の中で同名の変数を変更すると関数の外でも変更されてしまいます
function hello(){
        name=$1
        echo $name
}

name=Tom
echo $name
hello Mike
echo $name #nameの値がhello関数の中で変更されます(Mikeになる)

function hello(){
        local name=$1 # 変数宣言の前にlocalを付けると変数を関数の中だけ専用利用できます
}
```

shell scirpt 18(readonly)

```
# 値を変更できないようにするには、readonlyを用いる

var=31
readonly var # readonlyを設定
var=41 # 値を変更するとエラーになる



function hello() {
        echo “hello world”
}
readonly –f hello # 関数をreadonlyに設定
```

shell scirpt 19(pid, trap)

例えば、以下の場合にシェルスクリプト内でプロセスIDを用いることがあります。
・特定のプロセスを落とす
・プロセスIDをファイルに記憶する

pid is **$$ # 自分のプロセスID**

trapコマンドでプログラム終了時の挙動を設定します

trap “echo exit command is detected” 0 **# プログラム終了時に実行されるコマンド**
trap “echo Exit “ 15 **# kill -15 でのプロセス終了時に実行されるコマンド**
trap “rm –f $file && echo file deleted” 0 2 **# 終了時、割込み時に実行される**

1: 再起動、2:割り込み(ctrl+c)、9:強制停止(trapは設定できない)、15:プロセスの終了

shell scirpt 20(debug)

bash –x ./hello.sh　(use debug mode)

#!/bin/bash –x　(use debug mode)

set –x　(use debug mode)

set +x　(end debug mode)

shell scirpt 基本 演習問題5(exam5)

演習問題4のsum,avg,min,maxを関数化しましょう

以下のような数値を1列に並べたファイルを作成する
1
2
:
・ユーザから標準入力(ファイル名)を受け、ファイルの存在を確認する
・さらに標準入力を受け取り、その標準入力がsumの場合はファイルの合計を計算
・avgの場合は平均値を計算
・minの場合は最小値
・maxの場合は最大値を計算
・exitで処理を終了する
(ただし、ファイルの中には1~100までの数値が入っており、必ず1つ以上の数値が入っている事は前提とする)

shell scirpt 基本 演習問題6(exam6)

ファイル操作する関数

1. コマンドから処理を選択(select)。
- listの場合、ファイル一覧表示。
- deleteの場合ファイル選択後削除=>削除するファイルを入力
- renameの場合ファイル名変更=>名前変更するファイルを入力 =>変更する名前を入力
- showの場合ファイルの中身表示=>表示するファイルを入力
- exitの場合処理を終了する。

shell scirpt 基本 演習問題7(exam7)

1-1000まで数字を1秒おきにファイルに記入するジョブを立ち上げる関数を作成する。start, stopを用いて実行、停止する。

1. 引数がstartの場合、ジョブを実行する(ファイルを上書く)
2. ただし1ですでにジョブが動いている場合は‘already running’とメッセージを送って終了する
3. 引数がstopの場合、ジョブを終了(kill -9)。ただし停止中の場合は‘Not running’と表示する
4. 引数がstatusの場合、ジョブが停止中か起動状態を確認プロセスIDとともに表示する