

# touch

## ファイルの更新時刻を更新する

- ☐ Linux (bash)
- ☐ FreeBSD (sh)
- ☐ Solaris (sh)

書式 touch [-オプション] [ファイル] ...

touch コマンドを実行すると、引数で指定した[ファイル]の更新時刻(タイムスタンプ)が現在の時刻に更新されます(図E)。引数で指定したファイルが存在しない場合は、その名前のサイズゼロのファイルが作成されます。

図E ファイル更新時刻を更新する

```
$ ls -l memo.txt          lsコマンドでmemo.txtの更新時刻を表示
-rw-r----- 1 guest  user   13668 Jan 16 07:23 memo.txt
                                更新時刻が表示される

$ touch memo.txt          touchコマンドでmemo.txtの更新時刻を更新
$ ls -l memo.txt          再度lsコマンドでmemo.txtの更新時刻を表示
-rw-r----- 1 guest  user   13668 Jan 16 21:23 memo.txt
                                更新時刻が現在の時刻になる

$ date                    念のため現在の時刻を表示
Sat Jan 16 21:23:32 JST 2038
```

# sed

## ファイルの中の文字列を置換する

- ☐ Linux (bash)
- ☐ FreeBSD (sh)
- ☐ Solaris (sh)

書式 sed [-オプション] [プログラム] [ファイル] ...

sed コマンドは、引数で指定した[ファイル]に対し[プログラム]で指定した置換などの処理を施し、その結果を標準出力に出力します(図F)。引数の[ファイル]を省略した場合は標準入力からの入力になります。sedのプログラムの詳細についてはsedのオンラインマニュアルを参照してください。

図F 文字列を置換する

```
$ sed 's/January/February/g' old.txt > new.txt  old.txtの中のJanuaryという文字列を
                                                    Februaryに置換して、new.txtに書き込む
```

## >第13章 配列

13.1 概要 .....	274
13.2 配列への代入と参照 .....	275
13.3 配列の一括代入と一括参照 .....	277
13.4 bash以外のシェルで配列を使う方法 .....	279

# シェル上での配列

シェルスクリプトの処理によっては、変数として配列を使いたくなるような場面もあるでしょう(図A)。bashではシェル変数として配列を使用することができます。一方、bash以外の、配列が使用できないシェルでも、evalコマンドを使って配列に近い動作をさせることができます。本章では、これらをまとめて、シェル上で配列を扱う方法について解説します。

図A 配列week[]を使って曜日を求める

配列	
week	
Sunday	week[0]
Monday	week[1]
Tuesday	week[2]
Wednesday	week[3]
Thursday	week[4]
Friday	week[5]
Saturday	week[6]

## 配列への代入と参照

>第13章 配列

### bashではシェル変数として配列も扱える



書式

代入 配列名 [添字]=値

参照 \${配列名[添字]}

例

```
array[3]='hello world' .....arrayという配列の添字3の要素にメッセージを代入
echo "${array[3]}" .....arrayの添字3に代入されている値を表示
```

#### 基本事項

冒頭の書式のように、シェル変数名を配列名とし、その右側に[添字]を付けることにより、配列を扱うことができます。添字は「0」以上の整数です。添字は算術式とみなされ、添字にシェル変数が使用される場合、その頭に\$記号を付ける必要はありません。配列の[値]を参照する場合は、\$の後の配列名と添字を必ず{ }で囲みます<sup>注1</sup>。

#### 解説

bashでは冒頭のような形式の配列が使えるため、シェルスクリプト上で配列を使用する必要が生じた場合に便利です。ただし、移植性の点ではevalコマンドを使って疑似的に配列を扱うようにしたほうがよいでしょう<sup>注2</sup>。

配列の添字のシェル変数の頭の\$は省略できるため、より直感的な記述ができますが、配列の参照時には必ず{ }を付けて、\${array[i]}のような形式にする必要があるため、記述はやや複雑になります。さらに、通常のシェル変数と同様、展開された配列の値がさらにパス名展開や単語分割されるのを避けるため、ダブルクォートで囲んで"\${array[i]}"と記述するのが基本になります。

#### 配列への代入と参照の例

実際に配列に値を代入し、参照して表示している様子を図Aに示します。このように、配列の添字は数字を直接記述するほか、添字をシェル変数にしたり、数式にしたりすることもでき、いずれも正しく代入と参照が行われていることがわかります。

注1 シェル変数の代入と参照の項(p.159)も合わせて参照してください。

注2 詳しくは「bash以外のシェルで配列を使う方法」(p.279)を参照してください。

図A 配列への代入と参照の例

```
$ array[2]=two          配列arrayの要素2に、twoという文字列を代入
$ echo "${array[2]}"    配列arrayの要素2を参照してechoで表示
two                    たしかにtwoと表示される
$ i=5                  シェル変数iに5を代入
$ array[i]=five        配列arrayの要素iに、fiveという文字列を代入
$ echo "${array[i]}"    配列arrayの要素iを参照してechoで表示
five                  たしかにfiveと表示される
$ echo "${array[i-3]}"  添字をi-3にすると
two                  たしかにtwoと表示される
```

## 配列の一括代入と一括参照

配列のすべての要素について値を一括代入したり一括参照したりできる



一括代入 配列名=( [ 値1 ... ] )  
一括参照 \${配列名[@]}

例 array=(one two three) ...array[0] array[1] array[2]にそれぞれone two threeを代入  
echo "\${array[@]}" .....arrayのすべての要素を一括して表示

### 基本事項

冒頭の一括代入の書式のように記述すると、配列の要素0から順に[値]が一括して代入されます。冒頭の一括参照の書式では、配列のすべての要素が一括して参照されます。

### 解説

配列を使用する際、配列の各要素に初期値を代入するには、一括代入の書式が便利です。また、配列のすべての値を一括して参照することも可能であり、この書式はすべての位置パラメータを参照する特殊パラメータ"\$@"に類似しています。参照時には"\$@"と同様にダブルクォートで囲んで"\${array[@]}"の形にし、パス名展開や単語分割を避けるのが基本です。

なお、現在配列に代入されている要素の数は\${#array[@]}として参照できます。

### 一括代入と一括参照の例

配列に値を一括代入し、一括参照している例を図Aに示します。このように配列を設定しておけば、0~6までの範囲の値が代入されたシェル変数todayを使って、配列weekから曜日を参照できます。

なお、配列への一括代入の際に「week=([0]=Sunday [1]=Monday)」のように明示的に添字を指定することもできます。この場合、添字は順不同に並べてよく、途中の番号が飛んでいても構いません。

図A 一括代入と一括参照の例

```
$ week=(Sunday Monday Tuesday Wednesday Thursday Friday Saturday)
                                                    配列weekに曜日を一括代入
$ echo "${week[@]}"                                配列weekのすべての値を一括表示
Sunday Monday Tuesday Wednesday Thursday Friday Saturday
$ today=3                                          シェル変数todayに3を代入
$ echo "${week[today]}"                          シェル変数todayを添字として配列weekを参照
Wednesday                                         たしかにWednesdayと表示される
```

### 参照

シェル変数の代入と参照(p.159)  
ダブルクォート" "(p.209)

bash以外のシェルで配列を使う方法(p.279)

## bash以外のシェルで配列を使う方法

bash以外のシェルでもevalコマンドを用いて配列と同様の処理ができる

- ☐ Linux (bash)
- ☐ FreeBSD (sh)
- ☐ Solaris (sh)

### 解説

配列の機能を備えていないシェルにおいても、配列の添字を変数名の一部に含ませることにより、通常のシェル変数だけで配列と同様の処理を行えます。ここで、添字が別のシェル変数に格納されている場合、その添字をパラメータ展開した結果を変数名の一部に取り込んだ上で、再度evalコマンドで解釈して動作を行う必要があります。evalコマンドを使用するため、変数の値をクォートする際には、その解釈が2回行われることに注意する必要があります。その記述は複雑になります。しかし、evalを使って配列を実現しておけば、bash以外のシェルでも動作する、移植性の高いシェルスクリプトになります。

### evalを使った配列への代入と参照

evalを使って配列と同様に代入と参照を行っている例を図Aに示します。ここでは、week[]という配列を表現するために、week\_0~week\_6までの変数名のシェル変数を配列とみなして使用します。わかりやすいようにweekと添字の間にアンダースコア( )を入れてありますが、これはなくてもかまいません。ここで、week[2]のように、添字が定数の配列要素については、単にweek\_2として代入や参照ができます。

week[today]のように、添字がシェル変数の場合にはevalを使います。図A①の「eval echo \"\`week\_\$today`\"」は、1回解釈されて「echo "week\_3"」となり、これが再度解釈されて実質的にweek[3]の値が表示されます。ここで、weekの頭の\$は、1回目には解釈されないよう、\`としています。さらに、1回解釈された状態でダブルクォートが残るよう、全体を\"と\"で囲んでいることに注意してください。

次の②「eval week\_\$today=\"W E D\"」の部分は、1回解釈されて「week\_3="W E D"」となり、2回目の解釈でweek\_3に文字列が代入されます。「W E D」という文字列にはスペースが含まれるため、1回目の解釈を避けるためにシングルクォートで囲んで「'W E D'」としたものを、さらに2回目の解釈を避けるために\"で囲んで「\"'W E D'\"」としています。

### for文を使った配列への一括代入

前述のweekの配列に曜日を代入する部分を、for文を使って記述するとリストAのようになります。ループ中、「eval week\_\$i="\$day"」の部分は、たとえば、iの値が0ならば、1回解釈されて「week\_0="\$day"」となります。ここで、シェル変数dayの値は、参照と同時に代入を行っているため、これ以上パス名展開などは行われず、さらにダブルクォートで囲む必要はありません。

## 配列の削除

配列は、シェル変数と同様にunsetして削除することができます。unsetは、配列全体に対しても、配列中の個別の要素に対しても行えます。

図Bは、配列の要素および配列全体をunsetしている例です。図B①のunsetコマンドの引数では、week[1]の[1]の部分が、パス名展開であると解釈されないように全体をシングルクォート(' ')で囲んでいます。これを忘れると、もしもカレントディレクトリに「week1」という名前のファイルが存在した場合に、week[1]がweek1とパス名展開されてしまいます。

同様に、図B②のweek[today]についてもシングルクォートで囲みます。配列の添字のシェル変数todayは、変数名のままunsetコマンドに渡されますが、これでかまいません。

図B③のように、配列名を引数にしてunsetを実行すれば配列自体が削除されます。

## 配列から配列への一括代入

ある配列の内容をそのまま別の配列に一括代入したい場合は、配列の一括代入の書式の右辺の( )の中に、配列の一括参照の"\${array[@]}"の書式をダブルクォートつきで用います。具体的には図Cのようになり、ここでは配列arrayの値を一括して配列saveに代入しています。代入後の配列の値をechoで表示して、不要なパス名展開や単語分割が行われていないことが確認できます。

図B 配列の削除

```
$ week=(Sunday Monday Tuesday Wednesday Thursday Friday Saturday)
                                     配列weekをセット
$ unset 'week[1]'                  ①week[1]の要素のみ削除
$ echo "${week[@]}"                配列weekを一括表示すると
Sunday Tuesday Wednesday Thursday Friday Saturday   たしかにMondayが削除されている
$ today=4                          シェル変数todayに4を代入
$ unset 'week[today]'              ②week[today]の要素を削除
$ echo "${week[@]}"                再度配列weekを一括表示すると
Sunday Tuesday Wednesday Friday Saturday            たしかにThursdayが削除された
$ unset week                       ③配列week自体を削除
$ echo "${week[@]}"                配列weekを一括表示すると
何も表示されない
```

図C 配列から配列への一括代入

```
$ array=('Hello World' '*' '???')   配列arrayに適当な値を代入
$ echo "${array[@]}"                配列arrayの内容を一括表示する
Hello World * ???                  値が正しいことを確認(パス名展開・単語分割は回避)
$ save=("${array[@]}")              配列arrayの内容を配列saveに一括代入
$ echo "${save[@]}"                 配列saveの内容を一括表示する
Hello World * ???                  値が正しいことを確認(パス名展開・単語分割は回避)
```

### 参照

特殊パラメータ "\$@" (p.167)      unset (p.131)

図A evalを使った配列への代入と参照

```
$ week_0=Sunday week_1=Monday week_2=Tuesday      week_0からweek_6までに値を代入
$ week_3=Wednesday week_4=Thursday
$ week_5=Friday week_6=Saturday
$ echo "$week_2"                                     week[2]などはweek_2として参照すればよい
Tuesday                                              正しく表示される
$ today=3                                             シェル変数todayに3を代入
$ eval echo "\$week_$today"                         ①week[today]はevalを使ってこのように参照する
Wednesday                                           たしかにWednesdayと表示される
$ eval week_$today='W E D'                          ②week[today]='W E D'に相当する代入を実行
$ eval echo "\$week_$today"                         再びweek[today]を参照
W E D                                              スペースも含め、正しく表示される
```

リストA for文を使った配列への一括代入

```
i=0 .....シェル変数iに0を代入
for day in Sunday Monday Tuesday Wednesday Thursday Friday Saturday.....;
.....配列に代入する値の分だけfor文でループ
do .....ループの開始
    eval week_$i='$day' .....week[i]=$dayに相当する代入を実行
    i=`expr "$i" + 1` .....シェル変数iに1を加える
done .....ループの終了
```

参照

eval(p.98)

## >第14章 シェルスクリプトの ノウハウ&定石