Bash の代入

- 今回 URL とかコードとか多い感
- ぜひ手元の Bash とか Wandbox で試しながら聴いてもらえれば
- HTML: coord-e.github.io/slide-coins20lt-assignments-in-bash/
- PDF: coord-e.github.io/slide-coins20lt-assignments-in-bash/slide.pdf

Bash って何

- シェル
- コマンドラインを開いてコマンドを叩くと思うんですが、それ
- `$SHELL --version`

- Bash 4 の話をします
- このスライドでは POSIX sh 互換かどうかに興味を持たない
  - というかポータビリティの話をしない
  - ポータビリティの話をしようとするとでも Nix で…って大声が出る

Bash 代入クイズ

それぞれ出力は何でしょう（レギュレーション: ドキュメントなどは見ていい）

```
echo "-- Q1a --"
var1=$(false)
echo $?

echo "-- Q1b --"
readonly var2=$(false)
echo $?

echo "-- Q1c --"
var3=1 echo $var3

questions="Q1a Q1b Q1c Q2a Q2b Q2c"

echo "-- Q2a --"
questions1=$questions
echo $questions1

echo "-- Q2b --"
readonly questions2=$questions
echo $questions2

echo "-- Q2c --"
builtin readonly questions3=$questions
echo $questions3
```

Bash 代入クイズ: 答え

https://wandbox.org/permlink/GvPbrLyrmuK5ZBBU

```
echo "-- Q1a --"
var1=$(false)
echo $?  # 1

echo "-- Q1b --"
readonly var2=$(false)
echo $?  # 0

echo "-- Q1c --"
var3=1 echo $var3  # 改行のみ

questions="Q1a Q1b Q1c Q2a Q2b Q2c"

echo "-- Q2a --"
questions1=$questions
echo $questions1  # Q1a Q1b Q1c Q2a Q2b Q2c

echo "-- Q2b --"
readonly questions2=$questions
echo $questions2  # Q1a Q1b Q1c Q2a Q2b Q2c

echo "-- Q2c --"
builtin readonly questions3=$questions
echo $questions3  # Q1a
```

そんなことある？

- あって
- ドキュメントから説明できる正しい挙動
- これらがなんでこうなるのかを含め解説する

基礎: 変数いろいろ

変数

```
$ a=hello
$ echo $a
hello
```

ローカル変数

```
$ echo $var1  # unset

$ function f() { var1="I'm in f"; echo $var1; }
$ f
I'm in f
$ echo $var1  # var1 leaks from f
I'm in f
```

```
$ echo $var2  # unset

$ function g() { local var2="I'm in f"; echo $var2; }
$ g
I'm in g
$ echo $var2  # var2 is (dynamically) local in g
```

ローカルとはいえ、C などのローカル変数と違い、動的スコープであることに注意

配列変数

```
$ array=(a b c)
$ echo ${array[0]}
a
$ array+=(d)
$ echo ${array[3]}
d
$ echo ${array[-2]}
c
```

数値変数

```
$ declare -i num=1000
$ echo $num
1000
$ num=num+500  # arithmetic expansion without $(( ))
$ echo $num
1500
```

```
$ not_num=1000
$ echo $not_num
1000
$ not_num=not_num+500  # this does not expand normal variables
$ echo $not_num
not_num+500
$ not_num=$(( not_num+500 ))  # arithmetic expansion is performed
$ echo $not_num
1500
```

読み取り専用変数

```
$ readonly CONSTANT=hello
$ echo $CONSTANT
hello
$ CONSTANT=1
bash: read-only variable: CONSTANT
```

`local` や `declare` に `-r` を渡すことで読み取り専用にすることもできます

普通の代入？

代入の構文

```
name=[value]
```

Shell Parameters (Bash Reference Manual)

name というのは `[a-zA-Z0-9_]+` で, value がなんだかはよくわからないけど多分 word

代入の構文は何？

> 3. Parses the tokens into simple and compound commands (see Shell Commands).
> Shell Operation (Bash Reference Manual)

assignment statement は…?

manpage に答えが

> A simple command is a sequence of optional variable assignments followed by blank-separated words and redirections, and terminated by a control operator.

"Simple Commands" section in bash(1)

Simple Command の構文（忖度）

> A simple command is a sequence of optional variable assignments followed by blank-separated words and redirections, and terminated by a control operator.

"Simple Commands" section in bash(1)

```
[name=[value] ...] [word1 word2 ...]
```

普通の代入はコマンドが省略されたもの

今回はリダイレクトは無視

Q1a

```
echo "-- Q1a --"
var1=$(false)
echo $?  # 1
```

これもそれで、コマンドが省略されている

`false` は常に失敗の exit code を返すコマンド、`$?` は最後のコマンドの exit code が入っている

コマンドが省略された Simple Command の挙動

1. （コマンド部分で expansion をする）
2. assignment statement の value 部分でいくつかの expansion をする
3. コマンド名が残らなかったら
    i. 現在の環境に変数を代入
    ii. 前半の変数の value 部分に command substitution が含まれていれば
        a. **最後の command substitution の exit code をこの Simple Command の exit code として使う**
    iii. 含まれていなければ 0
4. （コマンド名が残ったら、…）

Simple Command Expansion (Bash Reference Manual)

command substitution というのは `$(command)` のことです

Q1a: まとめ

```
echo "-- Q1a --"
var1=$(false)
echo $?  # 1
```

- 最後の command substitution `$(false)` の exit code が 1 なので
- めでたし

Q1b

```
echo "-- Q1b --"
readonly var2=$(false)
echo $?  #0
```

- 前半に代入がないので代入部分はなし
- コマンド部分に `readonly var2=$(false)` が入る

代入が省略された Simple Command の挙動

1. コマンド部分で expansion をする
2. （assignment statement の value 部分でいくつかの expansion をする）
3. （コマンド名が残らなかったら）
4. コマンド名が残ったら
    i. 変数を代入した状態で §3.7.2 に沿ってコマンドの実行をする
    ii. 現在の環境は変更しない

Simple Command Expansion (Bash Reference Manual)

コマンドの実行

コマンド名（この場合 `readonly`）に

1. `/` が含まれてなければ
    i. 関数を探し、あったらそれを実行（この場合、ない）
    ii. ビルトインがあれば、それを実行
    iii. （なければ、 `$PATH` を探す）
2. （`/` が含まれていれば普通にそのファイルを実行）

Command Search and Execution (Bash Reference Manual)

`readonly` ビルトインの挙動

> The return status is zero unless an invalid option is supplied, one of the name arguments is not a valid shell variable or function name, or the -f option is supplied with a name that is not a shell function.

Bourne Shell Builtins (Bash Reference Manual)

どれでもないので 0 が返る

Q1b: まとめ

```
echo "-- Q1b --"
readonly var2=$(false)
echo $?  # 0
```

- 単に `readonly` が 0 を返す
- 代入ではなくコマンド実行なので単純な代入と挙動が違う
- めでたし
- `set -e` とか使っている場合注意する必要がある
  - `local`, `declare` なども同様

Q1c

```
echo "-- Q1c --"
var3=1 echo $var3  # 改行のみ
```

- 代入とコマンドが両方ある
- 代入部分: `var3=1`
- コマンド: `echo` `$var3`

Simple Command の挙動

1. **コマンド部分で expansion をする**
2. assignment statement の value 部分でいくつかの expansion をする
3. （コマンド名が残らなかったら）
4. コマンド名が残ったら
    i. 変数を代入した状態で §3.7.2 に沿ってコマンドの実行をする
    ii. 現在の環境は変更しない

Simple Command Expansion (Bash Reference Manual)

1. `var3=1 echo $var3`
2. `var3=1 echo`
3. `echo` を実行（`var3=1` の状態で）
4. 改行が出てくる

Q1c: まとめ

```
echo "-- Q1c --"
var3=1 echo $var3  # 改行のみ
```

- 先にコマンド部分を展開するので `$var3` が意図した値にならない
- めでたし
  - めでたくないが
  - びっくり

サブステージ: Word splitting について

クオートされていない展開結果がスペースとかで分割される

```
$ files="a b.txt c.txt"
$ cat $files
cat: a: No such file or directory
cat: b.txt: No such file or directory
cat: c.txt: No such file or directory
$ cat "$files"
cat: 'a b.txt c.txt': No such file or directory
```

スペースとかというのは `IFS` 変数の内容のことです

Q2a

```
questions="Q1a Q1b Q1c Q2a Q2b Q2c"

echo "-- Q2a --"
questions1=$questions
echo $questions1  # Q1a Q1b Q1c Q2a Q2b Q2c
```

代入のみ

1. `questions1=$questions`
2. `questions1=Q1a Q1b Q1c Q2a Q2b Q2c`
3. `questions1=Q1a` `Q1b` `Q1c` `Q2a` `Q2b` `Q2c`
   になりそうなものだが、ならない

Assignment statement で起きる expansion

1. tilde expansion
2. parameter expansion
3. command substitution
4. arithmetic expansion
5. quote removal

Simple Command Expansion (Bash Reference Manual)

> Word splitting is not performed, with the exception of "$@" as explained below.

Shell Parameters (Bash Reference Manual)

説明の順番ミスった感あるけど expansion については次回の LT でまとめようかな

Q2a: まとめ

```
questions="Q1a Q1b Q1c Q2a Q2b Q2c"

echo "-- Q2a --"
questions1=$questions
echo $questions1  # Q1a Q1b Q1c Q2a Q2b Q2c
```

- assignment の右辺では word splitting は起きない
- `var=$(command)` とかもクオート不要

Q2b

```
questions="Q1a Q1b Q1c Q2a Q2b Q2c"

echo "-- Q2b --"
readonly questions2=$questions
echo $questions2  # Q1a Q1b Q1c Q2a Q2b Q2c
```

コマンド: `readonly` `questions2=$questions`

?

- 今度は assignment statement とかではなくコマンド引数なので普通に word splitting が起きそう
- `readonly questions2=$questions`
- `readonly questions2=Q1a Q1b Q1c Q2a Q2b Q2c`
- `readonly questions2=Q1a Q1b Q1c Q2a Q2b Q2c` ?
  - にはならないんですよね
- すぐ嘘

```
# 並べるといかにびっくり挙動かわかる
$ cat questions2=$questions
$ readonly questions2=$questions
```

実装どうしてるんだよ

```
/* This is a hack to suppress word splitting for assignment statements
   given as arguments to builtins with the ASSIGNMENT_BUILTIN flag set. */
static void
fix_assignment_words (words)
```

bash-4.4/execute_cmd.c

- expansion 前に、代入をするビルトインの引数の代入の形をした部分について word splitting などをしないフラグを立てている
- 😇

Q2b: まとめ

```
questions="Q1a Q1b Q1c Q2a Q2b Q2c"

echo "-- Q2b --"
readonly questions2=$questions
echo $questions2  # Q1a Q1b Q1c Q2a Q2b Q2c
```

- めちゃくちゃアドホックなことやって代入と挙動を合わせてくる
  - word splitting が起きない
- 何？

Q3b

```
questions="Q1a Q1b Q1c Q2a Q2b Q2c"

echo "-- Q2c --"
builtin readonly questions3=$questions
echo $questions3  # Q1a
```

- コマンド: `builtin` `readonly` `questions3=$questions`
- これは普通に、マジで普通に word splitting する 😍
  - `builtin` `readonly` `questions3=Q1a Q1b Q1c Q2a Q2b Q2c`
  - `builtin` `readonly` `questions3=Q1a` `Q1b` `Q1c` `Q2a` `Q2b` `Q2c`
    - `readonly` を `questions3=Q1a` `Q1b` `Q1c` `Q2a` `Q2b` `Q2c` で実行
  - `questions3` に入るのは `Q1a` だけ

まとめ

- 昔の人すぐ激ヤバトークン置換
- 統一感のある挙動を期待するとマジでびっくりしてしまう
- なんでこんな最悪が…
- ShellCheck で戦いましょう

参考文献

- GNU Bash Manual
  - `info bash` でみれるやつ
- BASH(1) - General Commands Manual
  - `man bash` でみれるやつ