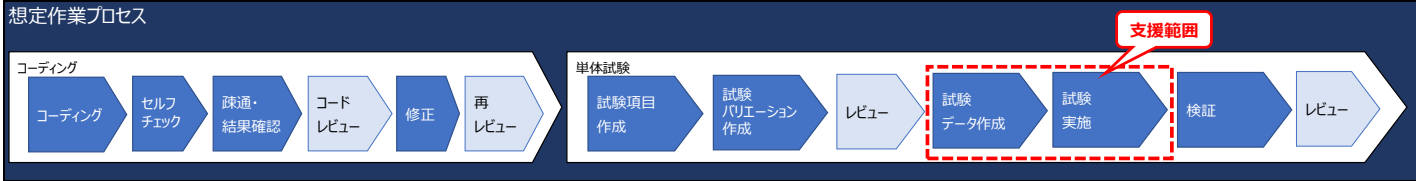
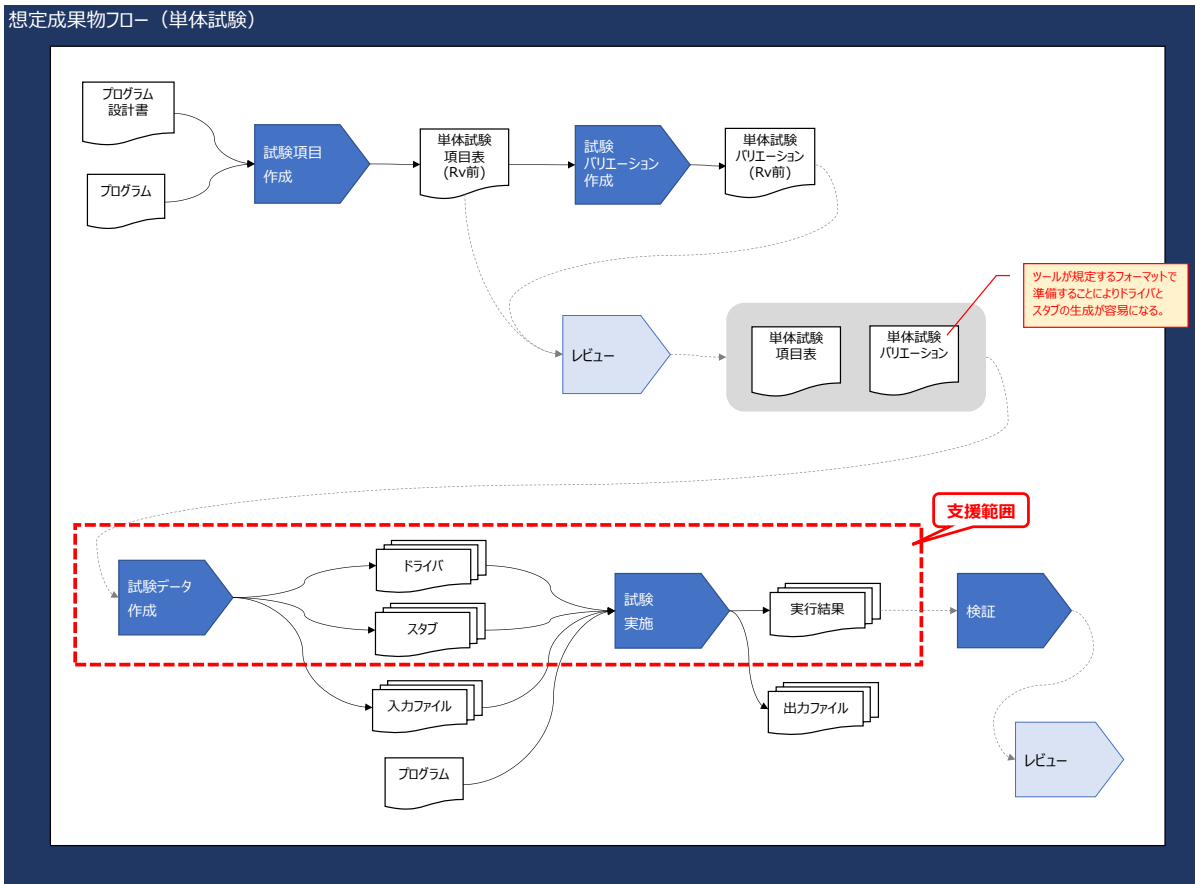


# 1 章. はじめに

- 1 - 1. 本資料の目的  
bashの単体試験を効率化するツール「Wfbash」の利用方法を示す。
- 1 - 2. 想定読者  
bashの開発をウォーターフォール型で行っており、これから単体試験作業を行う予定の開発者。
- 1 - 3. 想定作業プロセス  
bashの開発を下記の流れで行う場合において、Wfbashは「試験データ作成」「試験実施」の作業を支援します。



- 1 - 4. 想定成果物フロー  
bashの開発において作成する成果物のフローを下記で定義した場合において、Wfbashを利用して単体試験のバリエーションを作成することにより、「ドライバ」および「スタブ」作成の効率化が期待できます。また、期待する結果が出なかった場合のデバッグ作業を支援する機能を提供します。



- 1 - 5. ツール採用時の考慮事項

bashの単体試験を効率化するためのツールとしてBatsやShellSpecがあります。  
普通は一般的に使われている上記OSSを利用することを推奨します。

ただウォーターフォール型の開発では「試験項目表」「試験バリエーション」を開発者が表にしてまとめ、その内容をレビューがチェックするプロセスが必要となっていると思います。  
このような場合、レビューで品質を確保した試験バリエーションでそのまま試験資材を準備できる本ツールが役に立つので、自プロジェクト・開発の状況を踏まえ利用するかどうかをご判断ください。

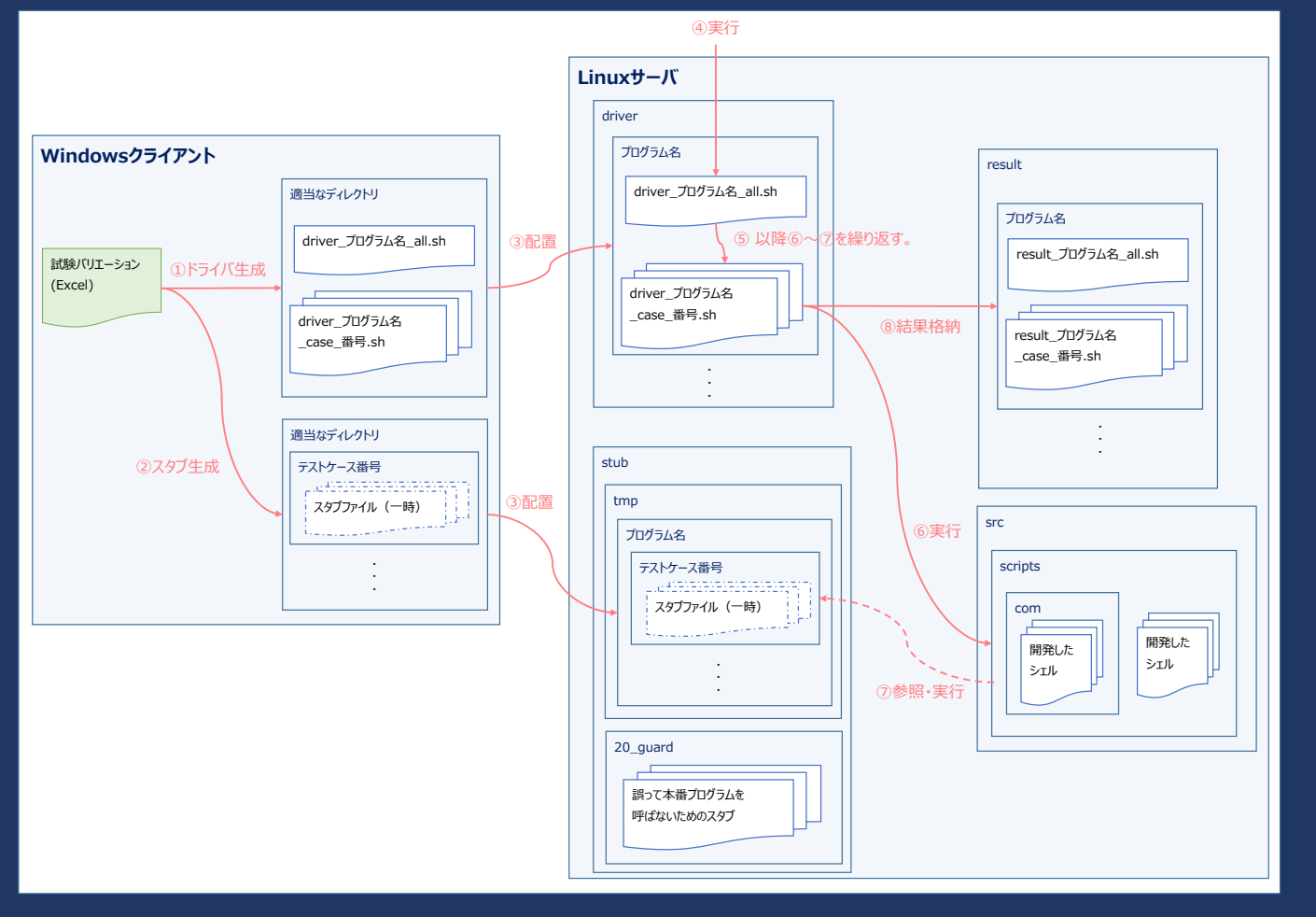
- 1 - 6. 本資料の構成

- |              |   |                                  |
|--------------|---|----------------------------------|
| 1 章. はじめに    | : | Wfbashを理解するうえで把握すべき前提・基本事項を説明します |
| 2 章. ツール概要   | : | 単体試験支援ツール「Wfbash」の概要を説明します       |
| 3 章. ツール利用方法 | : | ツールの利用方法を説明します                   |

## 2 章. ツール概要

WFbashを用いて試験をする際の概要と機能配置を以下に示します。

### ツール概要



ツールは以下の流れで利用することを想定しています。  
具体的な内容については3章をご覧ください。

#### ツールを利用した試験の流れ

- 3-1. 試験バリエーション作成
  - (1) コマンドのバリエーション投入
  - (2) スタブのバリエーション投入
  - (3) 期待値の投入
- 3-2. 試験資材作成
  - (1) 設定値の投入
  - (2) ドライバ・スタブの作成
  - (3) ドライバ・スタブの配置
  - (4) その他資材の準備
- 3-3. 試験実施
  - (1) 試験実施
  - (2) デバッグ機能利用
- 3-4. 検証 (カバレッジ確認)
  - (1) 設定値の投入
  - (2) カバレッジ情報生成

## 3章. ツール利用方法

### 3-1. 試験/バージョンの作成

試験/バージョンを作成する際は、同梱されている「シェル\_試験/バージョン\_xxx.xlsm」を雛型としてください。ファイル名は変更しても動作影響はありません。  
該当のExcelを開いたらシート「試験/バージョン」に試験内容を整理するための手順を説明します。

#### (1) コマンドのバージョン投入

実行パラメタ (M～V列) にコマンドのバージョンを投入します。

3行目にコマンド実行時に引数として渡す値の名称 (オプション名) を、4行目にスタブを「関数」と「ファイル」のどちらで準備するか指定してください。

3行目にオプション名を設定。  
例えば「-a 値 -i ファイル名」といった指定が可能なら下記の入力内容となります。

期待値	1	2	3	4	5	6	7	8	9	10	スタブ	1	2	3	4	5	6	7	8	9	10
1	-a	-f	file								com_a.sh	com_b.sh	date	./sample2.sh							
2	-オプション	-オプション(値なし)	-なしオプション								ファイル	ファイル	関数	関数							
3	0										0	0	0	0							
4	1	b	設定	tes							0	1	2	3							
5	9	1	設定	arere							9	8	6	7							
6	8	1		bbbb									4	4							
7	3	aaa		cccc																	

4行目にオプションの種類を設定します。  
下記の3種類から選択できます。  
-オプション: 「オプション名 値」として引数を指定するもの  
-オプション (値なし): 「オプション名」の実を引数として渡すため、値がないもの  
-なしオプション: 値のみ引数として指定するもの

試験/バージョン毎に設定したい値を投入する

#### (2) スタブのバージョン投入

スタブ (W～Z列) にスタブのバージョンを投入します。

3行目にスタブの名称を、4行目にスタブの種類を指定してください。

3行目にスタブ名を設定。プログラムで実行するコマンドの内容を設定すること。  
例えばsample2.shをカレントディレクトリで実行するなら./sample2.shが名称となる。

期待値	1	2	3	4	5	6	7	8	9	10	スタブ	1	2	3	4	5	6	7	8	9	10
1	-a	-f	file								com_a.sh	com_b.sh	date	./sample2.sh							
2	-オプション	-オプション(値なし)	-なしオプション								ファイル	ファイル	関数	関数							
3	0										0	0	0	0							
4	1	b	設定	tes							0	1	2	3							
5	9	1	設定	arere							9	8	6	7							
6	8	1		bbbb									4	4							
7	3	aaa		cccc																	

4行目にスタブの種類を設定します。  
- ファイル: スタブをファイルとして生成する場合に指定  
- 関数: スタブを関数として生成する場合に指定  
本指定を行った場合、ドライバにて3行目で指定した名称で関数が構築される。主にOSコマンドのスタブ生成時に利用。

試験/バージョン毎に設定したい値を投入する

※スタブを11個以上作りたい場合、AF列の右側の非表示列を再表示してください。

#### (3) 期待値の投入

L列にバージョン毎の終了ステータスの期待値を投入します。

#### (補足)

ツールが動作しなくなるため、列の挿入・削除は実施しないようにお願いします。

K列より左、およびBU列より右側はユーザ領域としているため、試験項目との紐づけ等、個別に投入したい情報があればそちらをご利用ください。

3-2. 試験資料作成

試験バージョンのレビューが完了した後、試験資料（ドライバ・スタブ）を生成するための手順を説明します。

(1) 設定値の投入

シート「設定」にある値を投入します。  
B列の「Linuxサーバでの格納先」に関する設定値は各プログラム共通で設定することを想定しています。

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										

ドライバ生成

スタブ生成

	設定値
実行シェル名称	sample1.sh
Windows上ファイル出力先	
作成ケース	開始
	終了
Linuxサーバでの格納先	実行シェル
	処理結果
	スタブ
	ガード用スタブ
	設定ファイル

B・C列のタイトルを参考に設定値を投入する。

←空の場合Excelと同じ場所へ出力

各プログラム共通の設定値を記述する。  
最初に値を決めてチーム内に展開すること。

(2) ドライバ・スタブの作成

作成したい対象に合わせてボタンを押下します。格納先は「Windows上ファイル出力先」に指定したフォルダに、フォルダ名「driver\_年月日\_時分秒」で作成されます。

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										

ドライバ生成

スタブ生成

スタブが生成されます。

ドライバが生成されます。

	設定値
実行シェル名称	sample1.sh
Windows上ファイル出力先	
作成ケース	開始
	終了
Linuxサーバでの格納先	実行シェル
	処理結果
	スタブ
	ガード用スタブ
	設定ファイル

←空の場合Excelと同じ場所へ出力

各プログラム共通の設定値を記述する。  
最初に値を決めてチーム内に展開すること。

(3) ドライバ・スタブの配置

ドライバとスタブをLinuxサーバ上に配置します。  
※注意：シート「設定」にて指定した「Linuxサーバでの格納先」に合わせて格納してください。なお文字コードは本ツールにてUTF8,BOMなしで作成されるため、変換不要です。  
またディレクトリについては自動生成されません。一部ディレクトリ（driver、stub/tmp、result）はプログラム名毎にサブディレクトリを事前に作成する必要があるので準備してください。（ディレクトリ毎の役割は2章.ツール概要参照）

(4) その他資料の準備

試験実行に必要な資料として「①入力ファイルの設置」と「②出力ファイルの取得」があります。  
「①入力ファイルの設置」についてはシェルの名称を「case\_試験バージョンの番号\_for\_Input」、  
「②出力ファイルの取得」についてはシェルの名称を「case\_試験バージョンの番号\_for\_Output」で準備すると  
試験バージョンに合わせて自動実行する作りとなっているため準備してください。  
不要な場合、対象処理はスキップするため準備不要です。

### (1) 試験実施

ドライブの実行を行う。「3-2. (3) ドライブ・スタブの配置」で資料を置いたフォルダへ移動後、下記コマンドを発行する。  
試験バリエーションで設定した番号毎に処理状況が表示されます。

```

: /torul/driver/sample1.sh$ ./driver_all_sample1.sh
test start...
caseNo=1 execute... end.
caseNo=2 execute... end.
caseNo=3 execute... end.
caseNo=4 execute... end.
caseNo=5 execute... end.
test end.

```

試験バリエーションの E x c e l のシート「設定」に記載した結果格納先に、処理結果が格納されます。  
結果のサマリはファイル名「result\_プログラム名\_all.txt」へ、  
試験バリエーション 1 つずつの結果は標準出力と標準エラー出力を合算した内容をファイル名「result\_プログラム\_case\_番号.txt」で格納します。

この場合、プログラム実行に関する処理で [実行プログラム名: ステップ番号:] 処理内容 が実行するステップ毎に出力されます。

```

./forUTL/driver/sample1.sh$ ./driver_all_sample1.sh debug
test start...
caseNo=1 execute...+ [./driver_case_1_sample1.sh:90:]rtnCode=8
+ [./driver_case_1_sample1.sh:91:]set +x
end.
caseNo=2 execute...+ [./driver_case_2_sample1.sh:90:]rtnCode=8
+ [./driver_case_2_sample1.sh:91:]set +x
end.
caseNo=3 execute...+ [./driver_case_3_sample1.sh:90:]rtnCode=8
+ [./driver_case_3_sample1.sh:91:]set +x
end.
caseNo=4 execute...+ [./driver_case_4_sample1.sh:90:]rtnCode=8
+ [./driver_case_4_sample1.sh:91:]set +x
end.
caseNo=5 execute...+ [./driver_case_5_sample1.sh:90:]rtnCode=8
+ [./driver_case_5_sample1.sh:91:]set +x
end.
test end.

```

```
result_sample1_sh_case_3.txt
9 replace ./sample2.sh to stab. rtn_code=7
10
11 ■実行コマンド確認
12 sample1.sh -a 1 -f arere
13
14 ■テスト実行
15 + [./driver_case_3_sample1.sh:66:] /home/kazuma/forUT/src/script/sample1.sh -a 1 -f arere
16 +++ [/home/kazuma/forUT/src/script/sample1.sh:4:] dirname /home/kazuma/forUT/src/script/sample1.sh
17 +++ [/home/kazuma/forUT/src/script/sample1.sh:4:] cd /home/kazuma/forUT/src/script
18 +++ [/home/kazuma/forUT/src/script/sample1.sh:5:] pwd
19 +++ [/home/kazuma/forUT/src/script/sample1.sh:5:] /home/kazuma/forUT/src/script/common_definition.sh
20 +++ [/home/kazuma/forUT/src/script/common_definition.sh:7:] dirname /home/kazuma/forUT/src/script/common_definition.sh
21 +++ [/home/kazuma/forUT/src/script/common_definition.sh:7:] export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/games:/usr/local/games:/snap/bin:/home/kazuma/forUT/stub/guard:/home/kazuma/forUT/stub/tmp/3:/home/kazuma/forUT/src/script/com
22 +++ [/home/kazuma/forUT/src/script/common_definition.sh:7:] PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/games:/usr/local/games:/snap/bin:/home/kazuma/forUT/stub/guard:/home/kazuma/forUT/stub/tmp/3:/home/kazuma/forUT/src/script/com
23 +++ [/home/kazuma/forUT/src/script/common_definition.sh:13:] dir_sysope_etc=/home/kazuma/forUT/01_src/etc
24 +++ [/home/kazuma/forUT/src/script/common_definition.sh:23:] warn_code=8
25 +++ [/home/kazuma/forUT/src/script/common_definition.sh:24:] error_code=0
26 +++ [/home/kazuma/forUT/src/script/sample1.sh:8:] com_a.sh
27 共通プログラムAを実行。正常終了
28 +++ [/home/kazuma/forUT/src/script/sample1.sh:9:] com_b.sh
29 /home/kazuma/forUT/src/script/sample1.sh: line 9: /home/kazuma/forUT/src/script/com/com_b.sh: Permission denied
30 ++ [/home/kazuma/forUT/src/script/sample1.sh:12:] date -aeras -f a
31 ++ [./driver_case_3_sample1.sh:66:] echo 'date is stab.'
32 date is stab.
33 ++ [./driver_case_3_sample1.sh:66:] echo ' |parm=-aeras -f a'
34 |parm=-aeras -f a
35 ++ [./driver_case_3_sample1.sh:66:] rtn_code=6
36 ++ [./driver_case_3_sample1.sh:66:] echo ' | 終了コード=6'
37 | 終了コード=6
38 ++ [./driver_case_3_sample1.sh:66:] return 6
39 ++ [/home/kazuma/forUT/src/script/sample1.sh:15:] /sample2.sh sample2向けの引数
40 ++ [./driver_case_3_sample1.sh:69:] echo './sample2.sh is stab.'
```

3-4. 検証 (カバレッジ確認)

(1) 設定値の投入

Excel「カバレッジ取得ツール.xlsm」を開きます。その後、シート「設定」に情報を投入します。

	A	B	C	D
1				
2				
3				
4				
5				
6				
7				
8				

作成

		設定値
格納先	ソース	C:\Users\%xxx%\Desktop\%試験ショット%\カバレッジ確認\src
	カバレッジ	C:\Users\%xxx%\Desktop\%試験ショット%\カバレッジ確認\result_all
Windows上ファイル出力先		C:\Users\%xxx%\Desktop\%試験ショット%\カバレッジ確認\処理結果

B・C列のタイトルを参考に設定値を投入する。

(2) カバレッジ情報生成

Linux上では、処理結果の格納先に「all」という名称のフォルダを指定すること、また、引数の1つめにdebugを指定することで、処理実行時に通ったステップの番号がディレクトリ「処理結果格納先/all」に、ファイル名「coverage\_プログラム名.txt」で出力されます。そのファイルを前手順のカバレッジの格納先として指定したフォルダへ、プログラムのソースファイルをソースの格納先として指定したフォルダへ格納します。その後、作成ボタンを押下することでカバレッジ情報が作成されます。

	A	B	C	
1				
2				
3				
4				
5				
6				
7				
8				

作成

		設定値
格納先	ソース	C:\Users\%xxx%\Desktop\%試験ショット%\カバレッジ確認\src
	カバレッジ	C:\Users\%xxx%\Desktop\%試験ショット%\カバレッジ確認\result_all
Windows上ファイル出力先		C:\Users\%xxx%\Desktop\%試験ショット%\カバレッジ確認\処理結果

作成ボタンを押下することで、カバレッジに格納されているリストの数だけソースフォルダにあるプログラムファイルを参考にカバレッジ情報を生成する