

# 俺の話を聞け！！LT大会 #9

2018/05/24(Thr)

## ExcelVBAから マルチプロセスでコマンド実行 『xxxx台連続ログインへの道』



会社名：非公開

発表者：井上 信之

# 自己紹介

## ▶ インフラ系の仕事

配布資料では  
非公開としています。

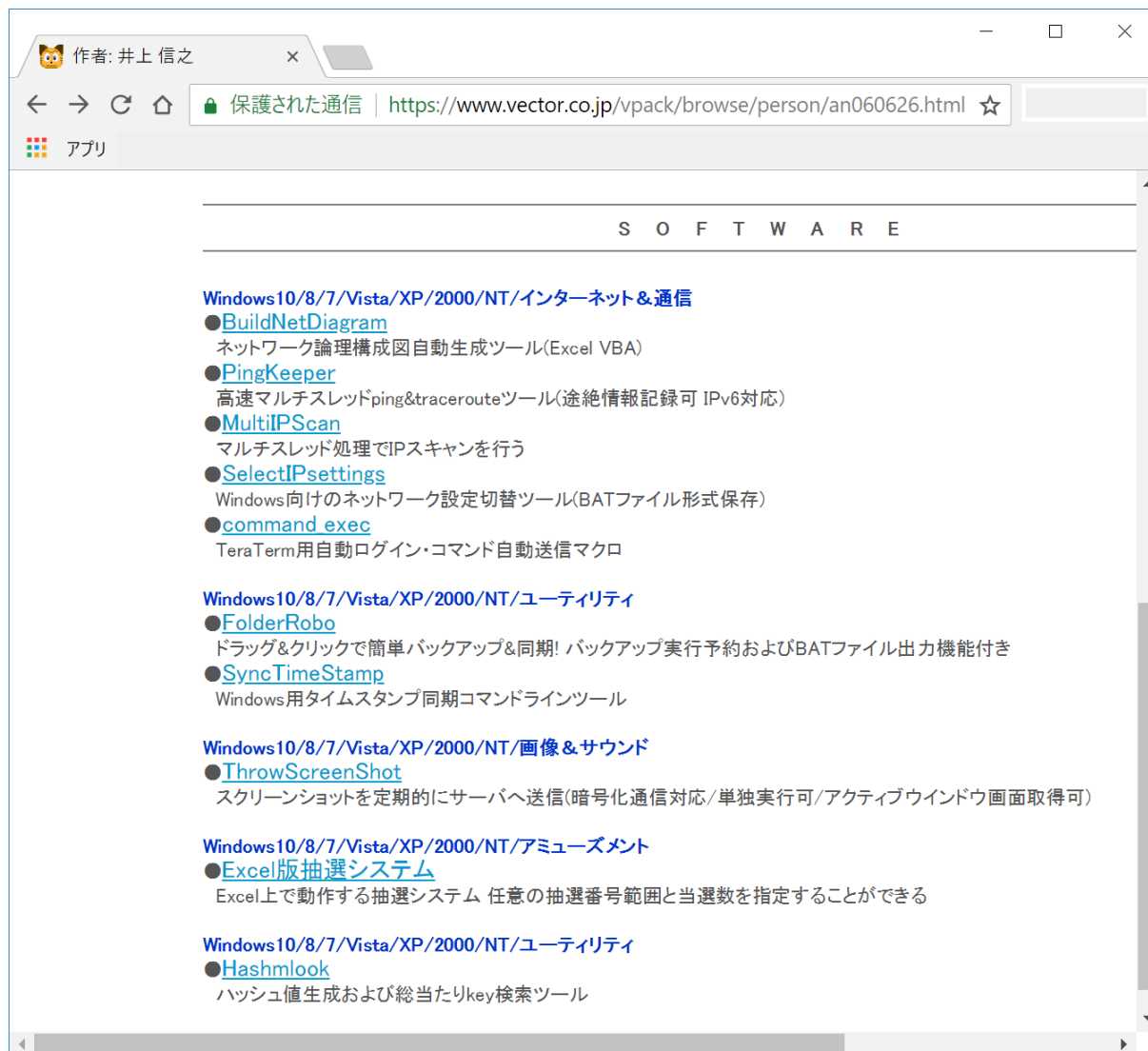
# 自己紹介（その2）

## ▶ 主な所有資格

配布資料では  
非公開としています。

# 公開ソフト一覧

<https://www.vector.co.jp/vpack/browse/person/an060626.html>



# コマンド連続実行のニーズ

- ▶ xxxx台のネットワーク機器のconfigを収集したい
- ▶ xxxx台のネットワーク機器の設定をまとめて変更したい
- ▶ xxxx台のネットワーク機器のポートの状態を収集したい

などなど、

同じような処理を1台ずつ直列ではやってもらえないとき...

# コマンドを並列に実行するには...

## Ping\_single.bat

```
ping 192.168.0.1  
ping 192.168.0.2  
ping 192.168.0.3
```

これだと、前のpingコマンドが終わるまで、次のpingコマンドは実行されません。

並列に実行するには、“start”コマンドの引数として、pingコマンドを指定します。

## Ping\_multi.bat

```
start ping 192.168.0.1  
start ping 192.168.0.2  
start ping 192.168.0.3
```

コマンド プロンプト(DOS窓)が3つ起動し、pingコマンドが3個同時に実行されます。

# コマンドを1,000個走らせたいときは...!?

- ▶ さすがに、BATファイルにコマンドをそのまま1000個並べるわけにはいかない。  
ブラクラ（ブラウザ・クラッシャー）状態になってしまう。
- ▶ 同時起動するコマンド数の上限を決め、上限を下回った場合に次のコマンドを起動させるようにすれば良い

# コマンド並列実行サンプル (Detune...いや、一般化バージョン)

本番環境用から、最低限の機能を取り出し、デチューン、いや、一般化したサンプルコードを、GitHubにて公開しています。

[https://github.com/gx3n-inue/multiExec\\_from\\_VBA](https://github.com/gx3n-inue/multiExec_from_VBA)

ただし、

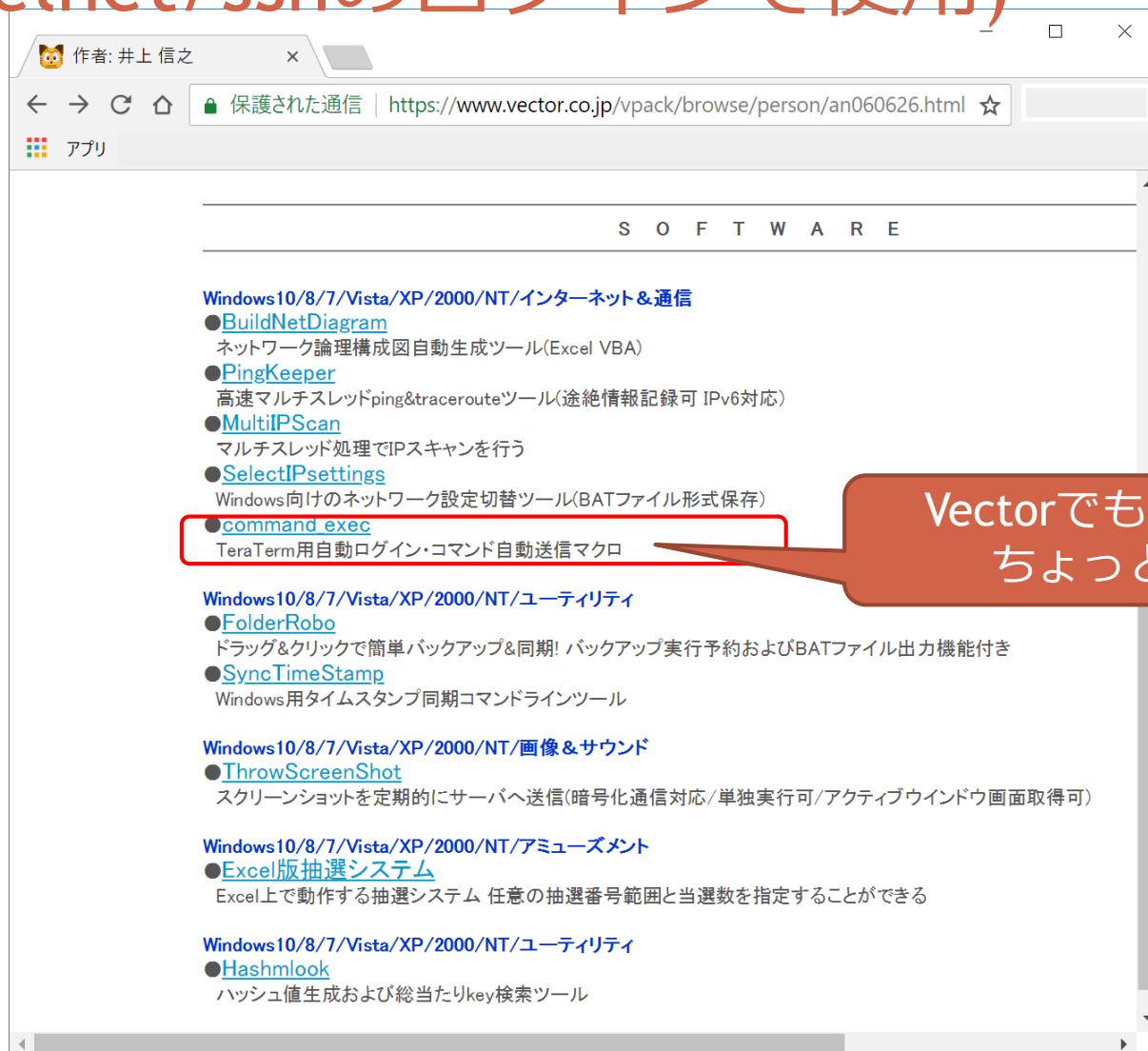
- ・ **緊急停止処理**がない(CTRL+Breakでマクロ停止)
- ・ コマンドの**実行に失敗しても無視**
- ・ **TeraTerm**のパス等も環境に合わせて変更が必要

なので、注意してください。

処理に成功したかどうかは、logファイルを調べ、コマンドの実行結果を調べる処理も必要。(本番環境ではそうしてる)



# 呼び出しているTeraTerm用マクロ (telnet/sshのログインで使用)



Vectorでも公開中(\*'艸`)  
ちょっと旧版だけど

# Excel VBAからの外部コマンドの実行方法

- ▶ Shell関数を利用する
  - ▶ MS-DOSの頃から存在するAPI
  - ▶ 戻り値はプロセスID
- ▶ WSH(Windows Script Host)オブジェクトを利用する
  - ▶ テキストファイルに記述したスクリプトを実行するスクリプト実行環境。Windows 98から搭載された。
  - ▶ COMを通じてレジストリの操作やWMIへのアクセスが可能であるなど、強力な機能を持っている。
  - ▶ WSHスクリプトを利用したウイルスや脆弱性攻撃などが増えたため、新しいスクリプト実行環境であるWindows PowerShellが登場した。

# VBAでの外部コマンドの起動1-1

## (Shell関数版 コマンドの終了を待つ)

### コマンドの終了を待つ場合

```
' 外部プログラムの実行
TaskId = Shell(Command_Str, WindowStyle))

' プロセスハンドルの取得
hProc = OpenProcess(PROCESS_ALL_ACCESS, 0, TaskId)

' プロセスハンドルが返されたかを判定
If hProc <> 0 Then
    ' プロセスのシグナル待ち
    Call WaitForSingleObject(hProc, INFINITE)
    ' プロセスクローズ
    CloseHandle hProc
End If
```

Shell関数実行後、WaitForSingleObjectプロシージャでプロセスの終了シグナルを待つ。

# VBAでの外部コマンドの起動1-2

## (Shell関数版 コマンドの終了を待たない)

### コマンドの終了を待つ場合

' 外部プログラムの実行

```
TaskId = Shell(Command_Str, WindowStyle))
```

' プロセスハンドルの取得

```
hProc = OpenProcess(PROCESS_ALL_ACCESS, 0, TaskId)
```

Shell関数実行後、そのまま次の処理に進めば良い。

# VBAでの外部コマンドの起動2-1

## (WSH(Execメソッド)版 コマンドの終了を待つ)

### コマンドの終了を待つ場合

```
Dim sh As New IWshRuntimeLibrary.WshShell ' WshShellクラスオブジェクト
Dim ex As WshExec ' Execメソッド戻り値

Set ex = sh.Exec(CmdStr(i))

' コマンドが終了まで待機する
Do While (ex.Status = WshRunning)
    Sleep 100
Loop
```

WSHShellクラスのExecメソッド実行後、ループ内でプロセスの終了シグナルを待つ。

Execメソッドでは出力結果は戻り値のWshExecオブジェクトに格納される。リダイレクトはそのまま利用できない。

標準出力を表示したい場合は、Runメソッドを利用する。

# VBAでの外部コマンドの起動2-2

## (WSH(Execメソッド)版 コマンドの終了を待たない)

### コマンドの終了を待つ場合

```
Dim sh As New IWshRuntimeLibrary.WshShell ' WshShellクラスオブジェクト
Dim ex As WshExec ' Execメソッド戻り値

Set ex = sh.Exec(CmdStr(i))
```

WSHShellクラスのExecメソッド実行後、そのまま次の処理に進めば良い。

# VBAでの外部コマンドの起動3-1

## (WSH(Runメソッド)版 コマンドの終了を待つ)

### コマンドの終了を待つ場合

```
Dim sh As New IWshRuntimeLibrary.WshShell ' WshShellクラスオブジェクト
```

```
sh.Run CmdStr(i), 1, true
```

- ▶ 2番目の引数はウィンドウの表示に関するオプションで1はデフォルト値
- ▶ 3番目の引数がtrueのときに終了を待つ。標準値はfalse

エラーコードは、きちんとプログラムの実行結果が返る。

しかし、プログラムの実行が終わるまで待機するので、並列処理には向かない。

# VBAでの外部コマンドの起動3-2

## (WSH(Runメソッド)版 コマンドの終了を待たない)

### コマンドの終了を待つ場合

```
Dim sh As New IWshRuntimeLibrary.WshShell ' WshShellクラスオブジェクト
```

```
sh.Run CmdStr(i), 1, false
```

- ▶ 2番目の引数はウィンドウの表示に関するオプションで1はデフォルト値
- ▶ 3番目の引数がtrueのときに終了を待つ。標準値はfalse

エラーコードは、きちんとプログラムの実行結果が返る。

しかし、プログラムの実行が終わるまで待機するので、並列処理には向かない。



# 並列実行メイン処理

## メイン処理

```
private sub mainLoop()  
    Dim i as long  
    Dim CmdStrs(0 to xx) as string  
  
    CmdStr(0) = "LIST001.BAT"  
    CmdStr(1) = "LIST002.BAT"  
    CmdStr(2) = "LIST003.BAT"  
    ...  
  
    For i = LBound(CmdStr) to UBound(CmdStr)  
        MyId = Shell(CmdStr[i], ウィンドウスタイル)  
    Next i  
  
End Sub
```

これだとブラウザクラッシャー状態になる

# 並列実行メイン処理

## メイン処理

```
private sub mainLoop()  
    Dim i as long  
    Dim CmdStrs(0 to 100) as string  
  
    CmdStr(0) = "LIST001.BAT"  
    CmdStr(1) = "LIST002.BAT"  
    CmdStr(2) = "LIST003.BAT"  
    ...  
  
    For i = LBound(CmdStr) to UBound(CmdStr)  
  
        ' 対象コマンドの起動プロセス数が上限を下回るまで待つ  
        wait_enableExec retryCountMAX, getInterval, i, pid_list  
  
        MyId = Shell(CmdStr[i], ウィンドウスタイル)  
  
    Next i  
  
End Sub
```

Shell関数を実行する前に、すでに起動して（まだ終了していない）対象コマンドのプロセス数をカウントし、上限以下になるまで待つようにすれば良い。

# 対象コマンドの起動プロセス数が 上限を下回るまで待つ処理

## メイン処理

```
Private Sub wait_enableExec(マクロ同時起動数上限数 As Long, PID取得リトライ回数 As Long, プロセス  
一覧取得間隔 As Long, i_Row As Long, ByRef pid_list() As Integer)
```

```
...
```

```
Do While True
```

```
    ' tasklist.exe コマンドの実行結果の取得
```

```
    Dim tasklistStr As String
```

```
    tasklistStr = get_TASKLIST_Resut()
```

```
    ' tasklist.exeの実行結果から対象プロセスのPIDを検索する
```

```
    For i = LBound(pid_list) To UBound(pid_list)
```

```
        If IsExist_targetProcesses(tasklistStr, pid_list(i)) Then
```

```
            pCount = pCount + 1
```

```
        Else
```

```
            pid_list(i) = 0
```

```
        End If
```

```
    Next i
```

```
    If pCount < マクロ同時起動数上限数 Then
```

```
        ' マクロ同時起動上限数を下回っている場合は、次の処理へ
```

```
        Exit Sub
```

```
    End If
```

ここで時間待ち処理

.....

```
Loop
```

# 実行中のPIDの取得方法

- ▶ WMI (Windows Management Information)を利用  
プロセス一覧を取得し、あらかじめ保存しておいた値と  
ProcessIDが一致するプロセスが存在するかをチェックする

# 指定したPIDのプロセスが存在しているか調べる

## 指定したPIDのプロセスが存在しているか調べる

```
Public Function IsExist_targetProcesses(target_pid As Integer) As Boolean
    (中略)
    ' WMI Win32_Process classのオブジェクトから PIDを検索する
    Dim Locator: Set Locator = CreateObject("WbemScripting.SWbemLocator")
    Dim Server: Set Server = Locator.ConnectServer
    Dim objSet: Set objSet = Server.ExecQuery("Select * From Win32_Process")
    Dim obj

    For Each obj In objSet
        If obj.ProcessID = target_pid Then
            ' 見つかった場合
            IsExist_targetProcesses_from_WMI = True
            Exit Function
        End If
    Next

    IsExist_targetProcesses_from_WMI = False

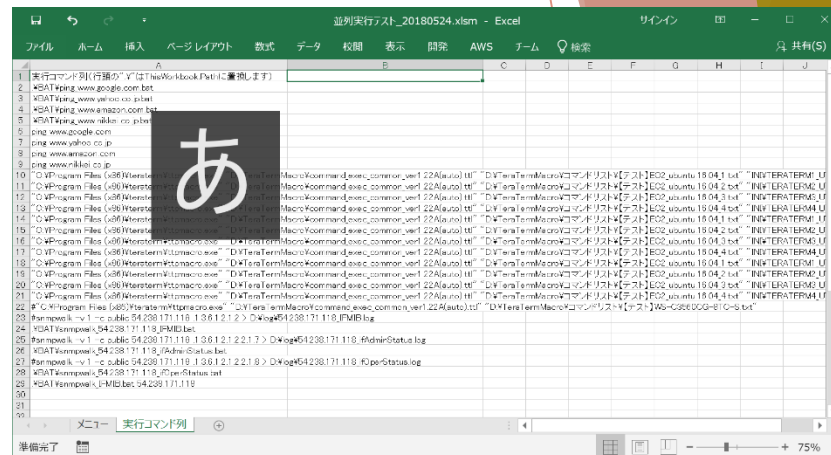
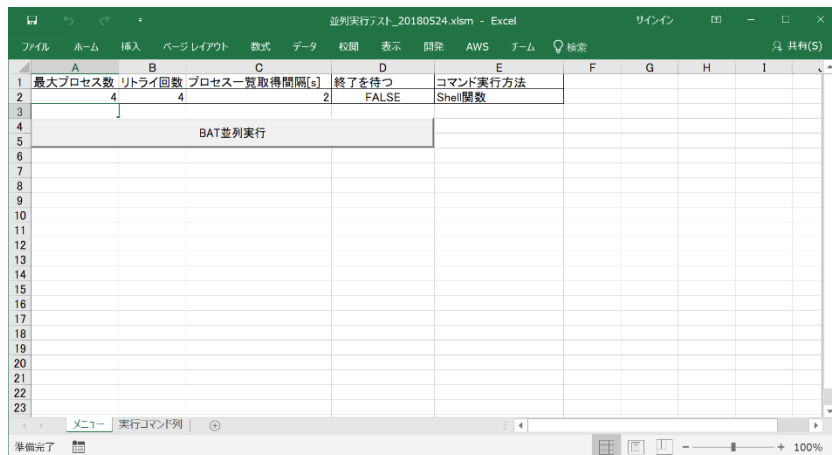
End Function
```

# それでは、実演...

- ▶ 接続先のNW機器のデモ用として、AWS EC2 Ubuntu Server 16.04 に連続ログインし、適当なコマンドを実行する。
- ▶ 実演用に、telnet, snmpを許可しています。（私のIPアドレスだけ）

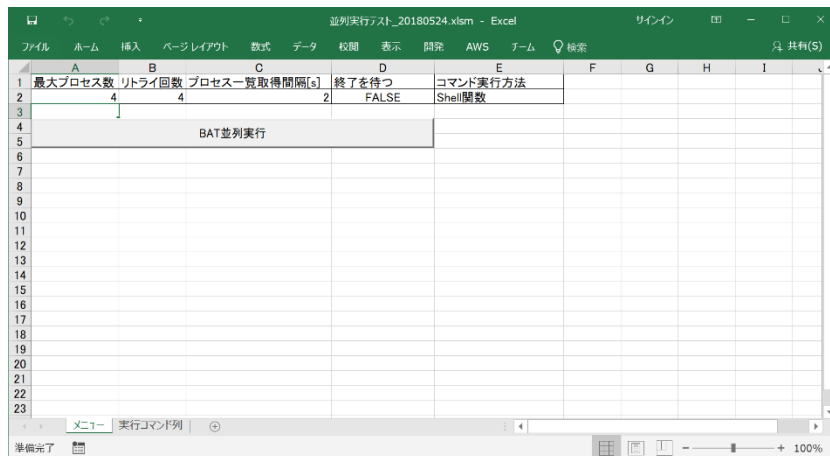
# 操作方法（その1）

「メニュー」と「実行コマンド列」の2つのシートがあります。



1. 「実行コマンド列」シートをクリックします。
2. A列に実行したいコマンドを列挙してください。空欄のセルまでが実行対象です。
  - IPアドレスはあくまでデモ時の値(AWS EC2の動的IPアドレス)なので、変更が必要です。
  - “#”から始まる行はコメント行として扱われ、実行されません。
  - 行頭の“.¥”は、ワークブックのパス(ThisworkBook.Path)に置換されます。
3. 「メニュー」シートをクリックします。

# 操作方法（その2）



4. 「最大プロセス数」、「リトライ回数」、「プロセス一覧取得間隔」に任意の値を入力してください。
5. 「プロセスの終了を待つ」は、FALSEだと並列実行、TRUEだと直列実行になります。
6. 「コマンド実行方法」は、“Shell関数”, “WSH(Run)”, “WSH(Exec)”から選べます。
  - Shell関数 Win32APIを使用。リダイレクトも利用可
  - WSH(Exec) リダイレクトは利用できません
  - WSH(Run) リダイレクトも利用可
7. 「BAT並列実行」ボタンをクリックします。

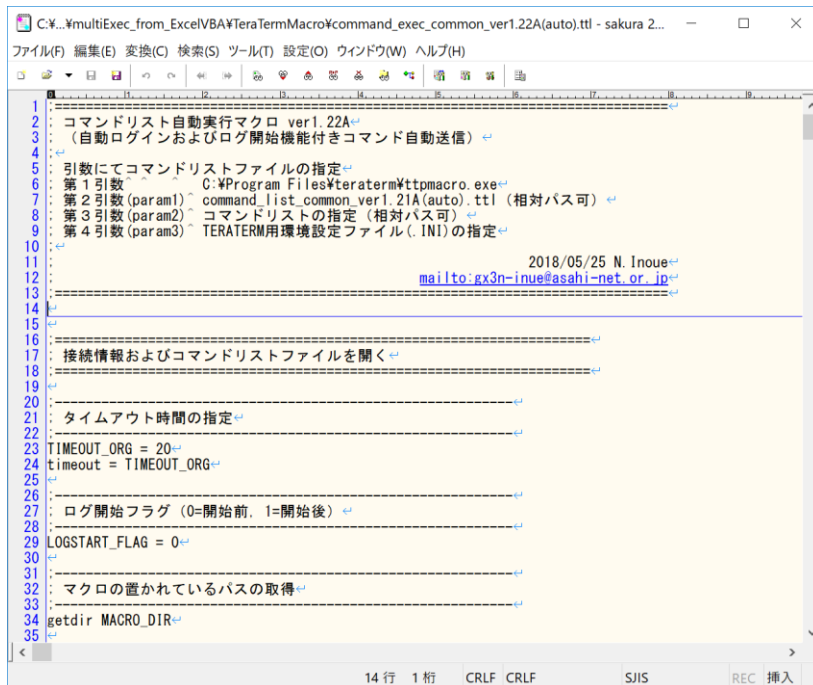


# 操作方法（その3）

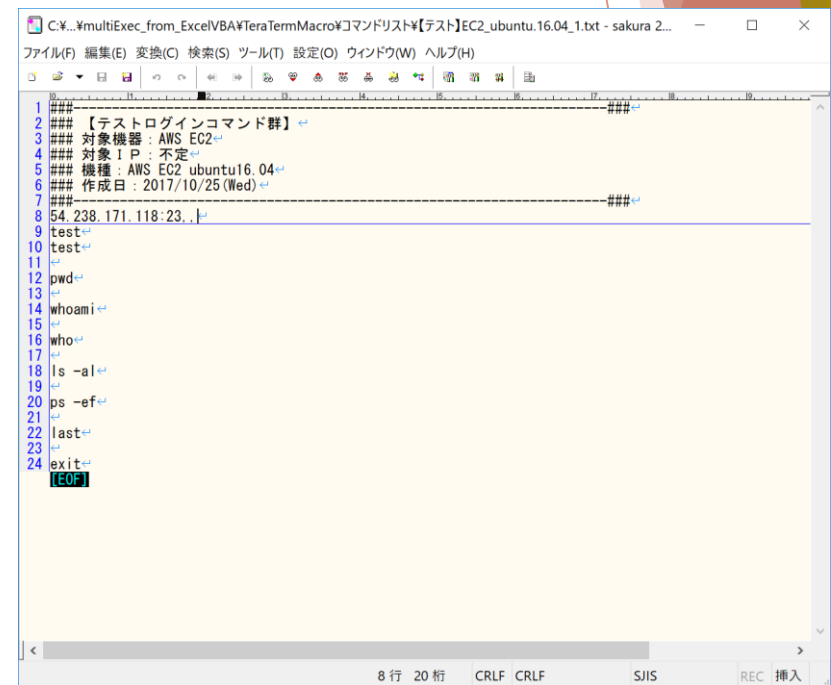
- ▶ NW機器およびLinux(telnet/SSHサーバ)への自動ログインおよびコマンド送信には、command\_exec(TeraTerm用自動ログイン・コマンド自動送信マクロ)を使用しています。

<https://www.vector.co.jp/soft/winnt/net/se516693.html>

- ▶ 詳しい利用方法は、readme.txtを読んでください。



```
1 コマンドリスト自動実行マクロ ver1.22A
2 (自動ログインおよびログ開始機能付きコマンド自動送信)
3
4 引数にてコマンドリストファイルの指定
5 第1引数 C:\Program Files\TeraTerm\TeraTermMacro\command_exec_common_ver1.22A(auto).ttl (相対パス可)
6 第2引数(param1) command_list_common_ver1.21A(auto).ttl (相対パス可)
7 第3引数(param2) コマンドリストの指定 (相対パス可)
8 第4引数(param3) TERATERM用環境設定ファイル(.INI)の指定
9
10
11 2018/05/25 N. Inoue
12 mailto:gx3n-inoue@asahi-net.or.jp
13
14
15 =====
16 接続情報およびコマンドリストファイルを開く
17 =====
18
19
20 タイムアウト時間の指定
21
22
23 TIMEOUT_ORG = 20
24 timeout = TIMEOUT_ORG
25
26
27 ログ開始フラグ (0=開始前, 1=開始後)
28
29 LOGSTART_FLAG = 0
30
31
32 マクロの置かれているパスの取得
33
34 getdir MACRO_DIR
35
```



```
1 #####
2 【テストログインコマンド群】
3 対象機器: AWS EC2
4 対象IP: 不定
5 機種: AWS EC2 ubuntu16.04
6 作成日: 2017/10/25 (Wed)
7 #####
8 54.238.171.118:23
9
10 test
11 test
12
13 pwd
14
15 whoami
16
17 who
18
19 ls -al
20
21 ps -ef
22
23 last
24
25 exit
26 [EOF]
```