

「コンピュータの舞台裏」

第2回

「業務への正規表現の応用」

2015-12-19

By くーへん

<https://github.com/flhtc1964/Computer>

(自己|事故) 紹介

- 1964年 東京都生まれ
- N88BASIC、MSDOS、VBA、VBS 、PHP、C#
- SunOS4.1.3、FreeBSD
Cシェル、grep、gawk、sed、Perl
- UUCP、NetNews、kermit
- 2000年 4月から大阪の某法律事務所へ転職
情報システム課に所属し会社更生システム(SQL
Server + Access[adp])の作成、データベース作成
&メンテナンスと、主にOffice製品等のヘルプデ
スクを担当現在に至る

BAT(バッチ)ファイルのおさらい

- 「対話形式」でコマンドを入力し処理するルーチンワーク(定型業務)を
事前にテキストファイルへ記述しておいて
必要な時に実行させる処理方法
- テキストファイル形式の拡張子は「. BAT」
#小文字で「.bat」でも可能

【注意】 Windowsの拡張子が非表示だと
見た目はfoo.batだが
実際はfoo.bat.txtに注意！

MS-DOSなら → 「コマンドプロンプト」 【 Unixなら → 「sh(シェル)」 】

- 対話形式でコマンドを入力して処理する道具

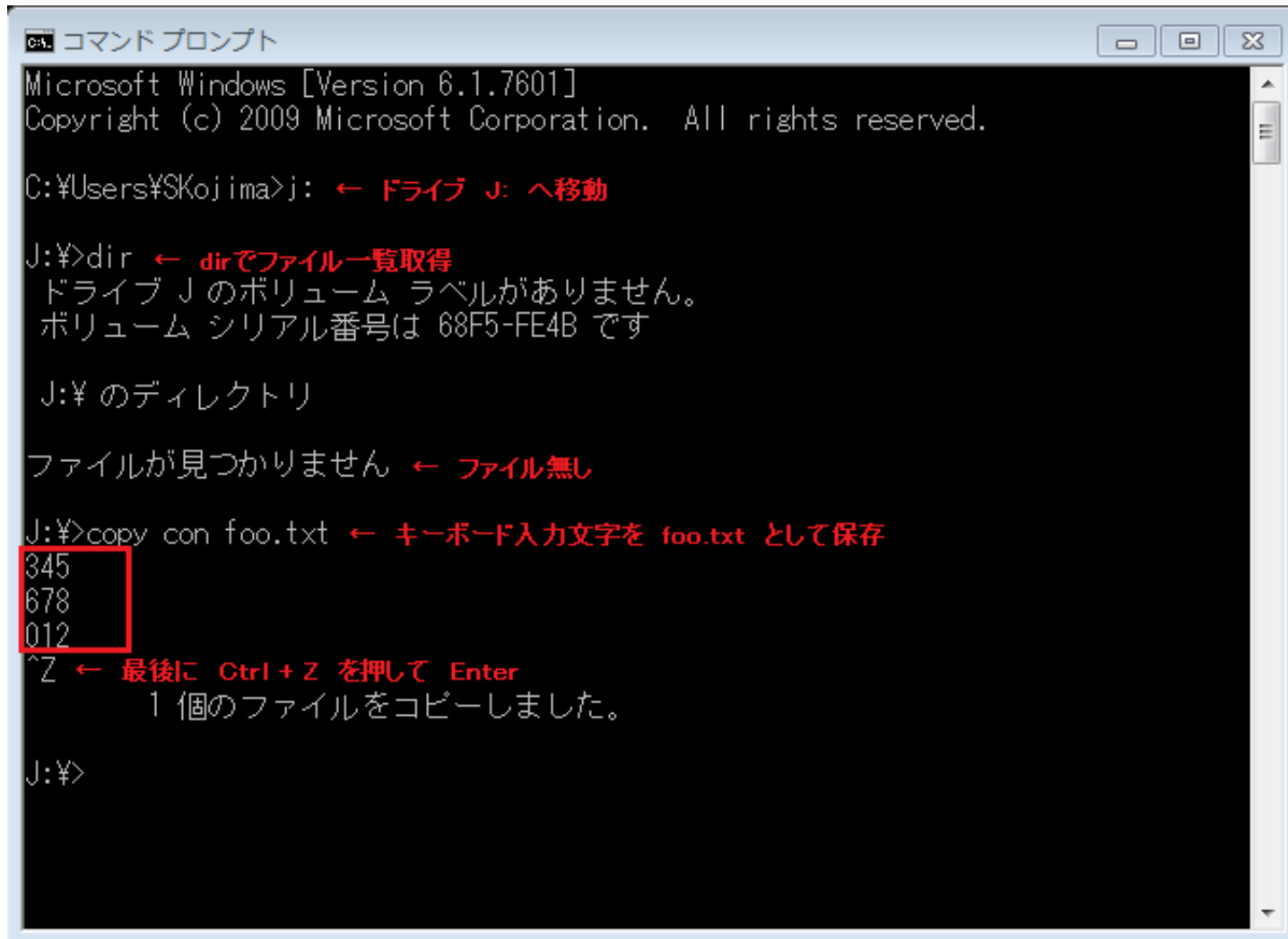
「リダイレクト → **<** 」 手入力せずファイルで入力
sort コマンドの利用

「リダイレクト → **>** 」 新規ファイル作成

「リダイレクト → **> >** 」 ファイルへ追加

「パイプ → **|** 」 フィルター

コマンドプロンプトでテキスト型 データファイルを作成してみる



```
C:\> コマンド プロンプト
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Skojima>j: ← ドライブ J: へ移動

J:\>dir ← dirでファイル一覧取得
ドライブ J のボリューム ラベルがありません。
ボリューム シリアル番号は 68F5-FE4B です

J:\のディレクトリ

ファイルが見つかりません ← ファイル無し

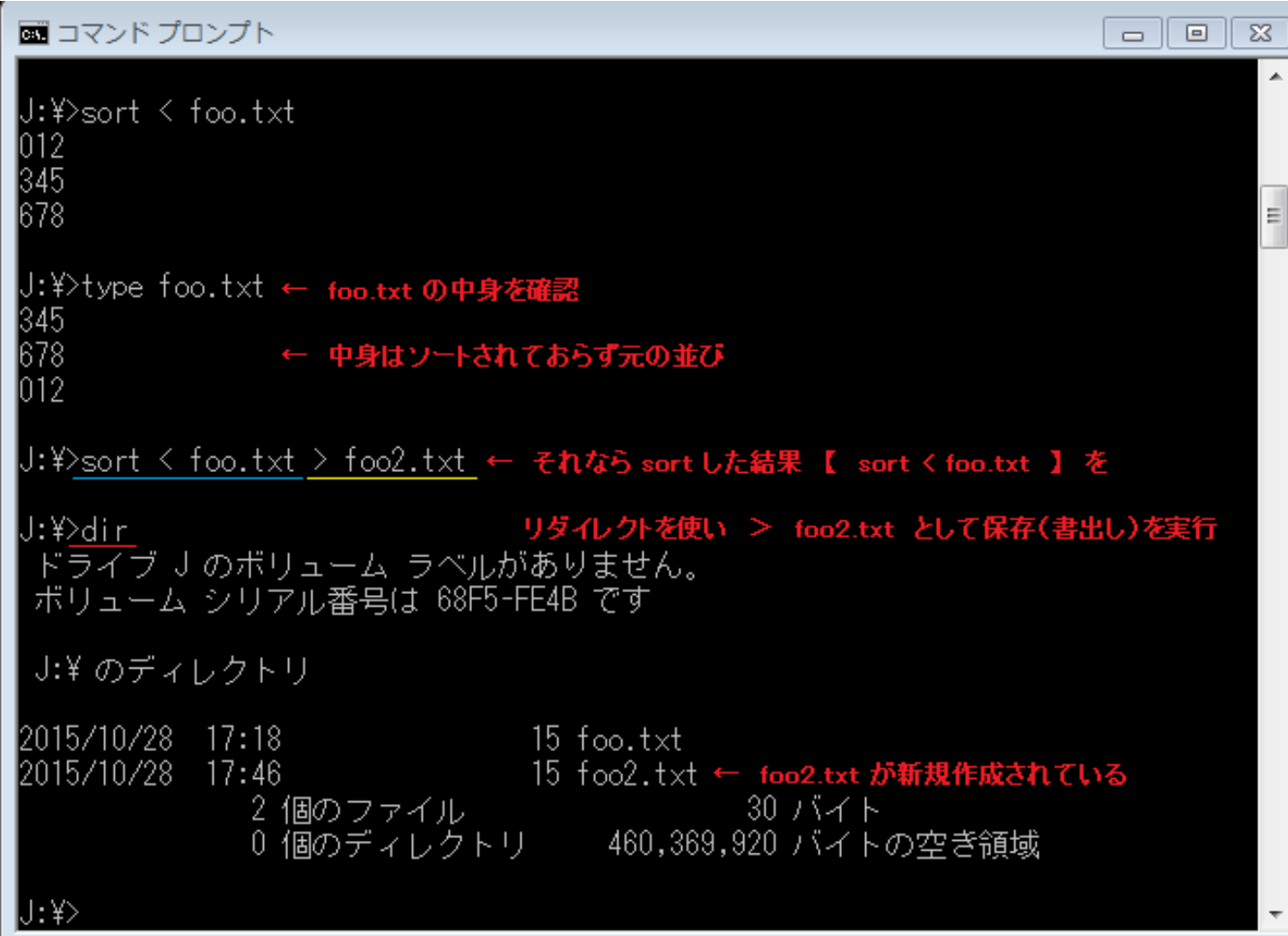
J:\>copy con foo.txt ← キーボード入力文字を foo.txt として保存
345
678
012
^Z ← 最後に Ctrl + Z を押して Enter
1 個のファイルをコピーしました。

J:\>
```

「リダイレクト→ < 」 手入力せずファイルで入力 sort コマンドの利用

```
012  
^Z  
1 個のファイルをコピーしました。 ← 1個のファイルが作成された  
J:¥>dir ← 再度 dir でファイル一覧取得  
ドライブ J のボリューム ラベルがありません。  
ボリューム シリアル番号は 68F5-FE4B です  
  
J:¥ のディレクトリ  
  
2015/10/28  17:18                15 foo.txt ← foo.txt が作成されている  
                1 個のファイル                15 バイト  
                0 個のディレクトリ            460,369,920 バイトの空き領域  
  
J:¥>type foo.txt ← type で中身を確認  
345  
678  
012  
  
J:¥>sort < foo.txt ← sort に対して foo.txt の中身をリダイレクト(データを送り込む)  
012  
345 ← 順序が昇順になって【 表示 】された  
678  
J:¥>
```

sort < foo.txt > foo2.txt
「リダイレクト→ > 」 新規ファイル作成



```
CA. コマンド プロンプト

J:¥>sort < foo.txt
012
345
678

J:¥>type foo.txt ← foo.txt の中身を確認
345
678
012
← 中身はソートされておらず元の並び

J:¥>sort < foo.txt <u>> foo2.txt ← それなら sort した結果 【 sort < foo.txt 】 を
リダイレクトを使い > foo2.txt として保存(書出し)を実行

J:¥>dir
ドライブ J のボリューム ラベルがありません。
ボリューム シリアル番号は 68F5-FE4B です

J:¥ のディレクトリ

2015/10/28 17:18          15 foo.txt
2015/10/28 17:46          15 foo2.txt ← foo2.txt が新規作成されている
                2 個のファイル              30 バイト
                0 個のディレクトリ        460,369,920 バイトの空き領域

J:¥>
```

コマンド プロンプト

J:¥ のディレクトリ

2015/10/28	17:18	15	foo.txt	
2015/10/28	17:46	15	foo2.txt	
		2 個のファイル		30 バイト
		0 個のディレクトリ		460,369,920 バイトの空き領域

J:¥>type foo*.txt ← type でワイルドカードを使い

foo.txt foo*.txt

を指定 foo.txt と foo2.txt が表示された

345
678
012

foo2.txt

012 ← foo2.txt はソート後のデータが保存されている

345
678

J:¥>

C:\ コマンドプロンプト

foo2.txt

012

345

678

J:¥>copy foo*.txt foo3.txt ← コピーしろ
foo.txt foo*.txt にマッチするファイルをまとめた後
foo2.txt foo3.txt として作成
1 個のファイルをコピーしました。

J:¥>dir
ドライブ J のボリューム ラベルがありません。
ボリューム シリアル番号は 68F5-FE4B です

J:¥ のディレクトリ

2015/10/28	18:08	15	foo.txt	
2015/10/28	17:46	15	foo2.txt	
2015/10/28	18:08	31	foo3.txt	← foo3.txt が作成された
		3 個のファイル		61 バイト
		0 個のディレクトリ		460,369,920 バイトの空き領域

C:\ コマンド プロンプト

J:\>type foo3.txt ← foo3.txt の中身を確認

345
678
012
012
345
678

J:\>copy foo2.txt + foo.txt foo4.txt ← 結合するファイルの順番を + で指定する
foo2.txt
foo.txt

1 個のファイルをコピーしました。

J:\>dir

ドライブ J のボリューム ラベルがありません。
ボリューム シリアル番号は 68F5-FE4B です

J:\> のディレクトリ

2015/10/28	18:08	15	foo.txt	
2015/10/28	17:46	15	foo2.txt	
2015/10/28	18:08	31	foo3.txt	
2015/10/28	18:14	31	foo4.txt	← foo4.txt が作成された
			4 個のファイル	92 バイト
			0 個のディレクトリ	460,369,920 バイトの空き領域

```
コマンドプロンプト
foo.txt
    1 個のファイルをコピーしました。

J:¥>dir
ドライブ J のボリューム ラベルがありません。
ボリューム シリアル番号は 68F5-FE4B です

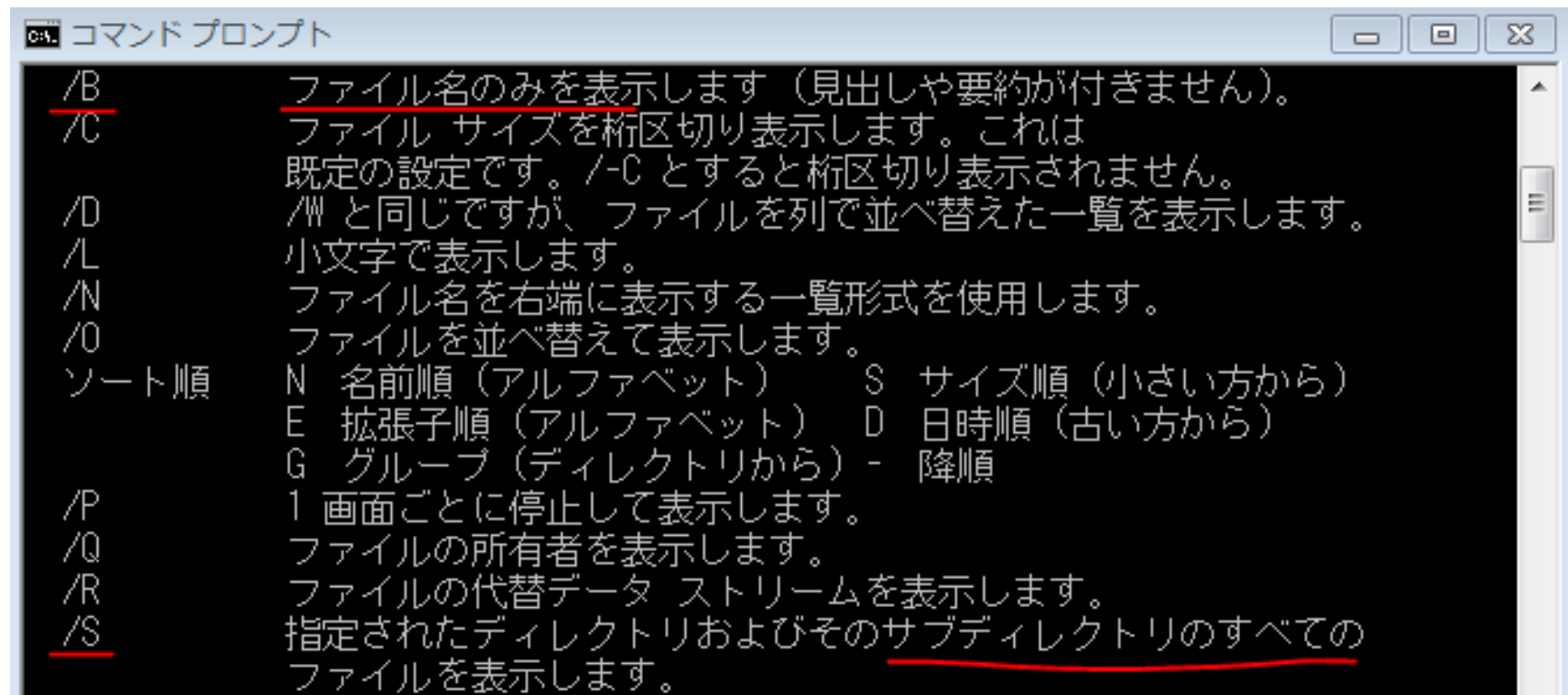
J:¥ のディレクトリ

2015/10/28  18:08                15 foo.txt
2015/10/28  17:46                15 foo2.txt
2015/10/28  18:08                31 foo3.txt
2015/10/28  18:14                31 foo4.txt
           4 個のファイル                92 バイト
           0 個のディレクトリ        460,369,920 バイトの空き領域

J:¥>type foo4.txt  ← foo4.txt の中身を確認
012
345
678
345
678
012
J:¥>
```

← ソートされている foo2.txt が先頭になっている

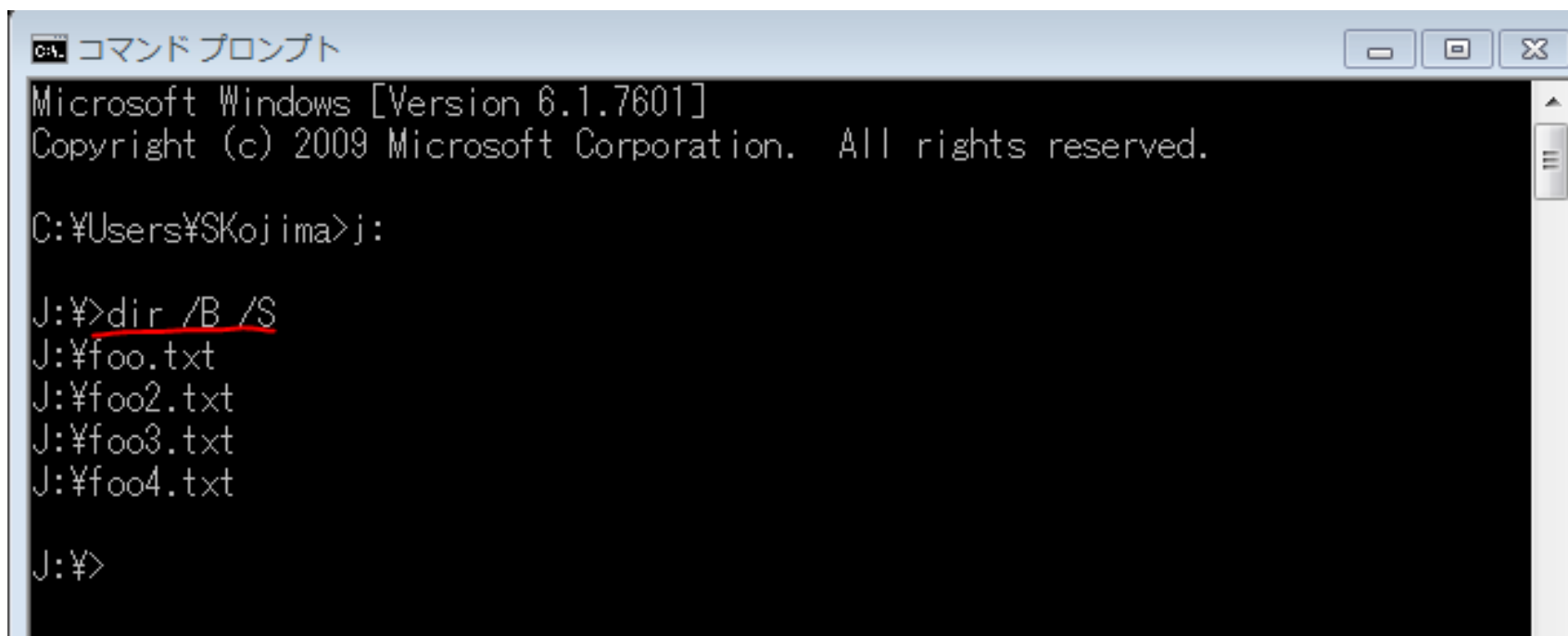
Dir /? コマンドヘルプを表示



コマンドプロンプト

```
/B      ファイル名のみを表示します（見出しや要約が付きません）。  
/C      ファイル サイズを桁区切り表示します。これは既定の設定です。  
          /-C とすると桁区切り表示されません。  
/D      /W と同じですが、ファイルを列で並べ替えた一覧を表示します。  
/L      小文字で表示します。  
/N      ファイル名を右端に表示する一覧形式を使用します。  
/O      ファイルを並べ替えて表示します。  
ソート順  N 名前順（アルファベット）      S サイズ順（小さい方から）  
          E 拡張子順（アルファベット）      D 日時順（古い方から）  
          G グループ（ディレクトリから） - 降順  
/P      1 画面ごとに停止して表示します。  
/Q      ファイルの所有者を表示します。  
/R      ファイルの代替データ ストリームを表示します。  
/S      指定されたディレクトリおよびそのサブディレクトリのすべての  
          ファイルを表示します。
```

dir /B /S コマンドでファイル一覧



```
コマンド プロンプト
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\S\Kojima>j:

J:>dir /B /S
J:\foo.txt
J:\foo2.txt
J:\foo3.txt
J:\foo4.txt

J:>
```

「リダイレクト→ > 」

dir /b /s > test.txt ファイル新規作成

```
cmd コマンドプロンプト

J:\>dir /b /s
J:\foo.txt
J:\foo2.txt
J:\foo3.txt
J:\foo4.txt
J:\test.txt

J:\>dir /b /s > test.txt

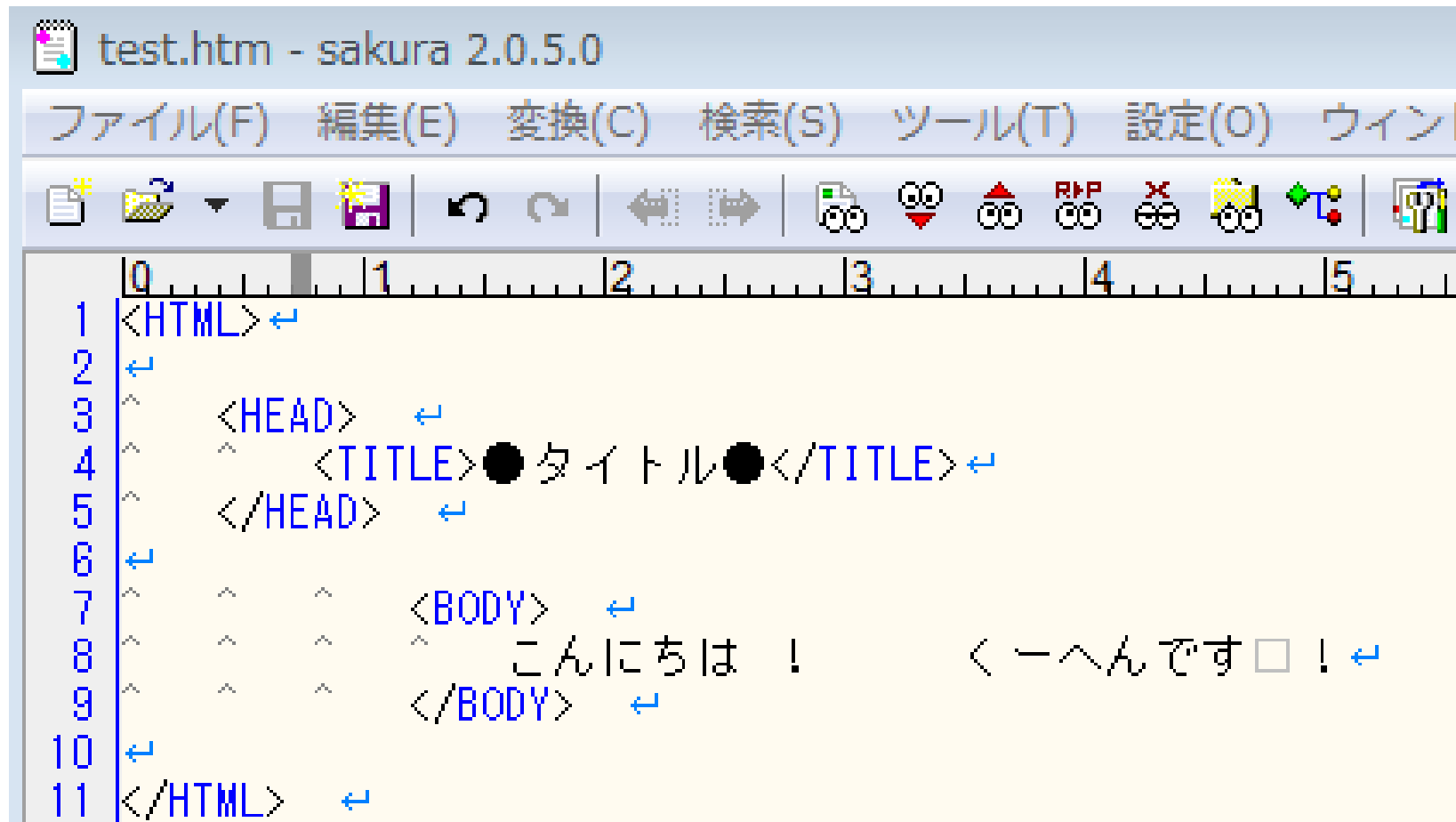
J:\>dir
ドライブ J のボリューム ラベルがありません。
ボリューム シリアル番号は 68F5-FE4B です

J:\ のディレクトリ

2015/10/28  18:08           15 foo.txt
2015/10/28  17:46           15 foo2.txt
2015/10/28  18:08           31 foo3.txt
2015/10/28  18:14           31 foo4.txt
2015/11/06  12:12           64 test.txt

                5 個のファイル                156 バイト
```

作成したtest.txtを使い最低限のHTML形式ファイルを自動作成する



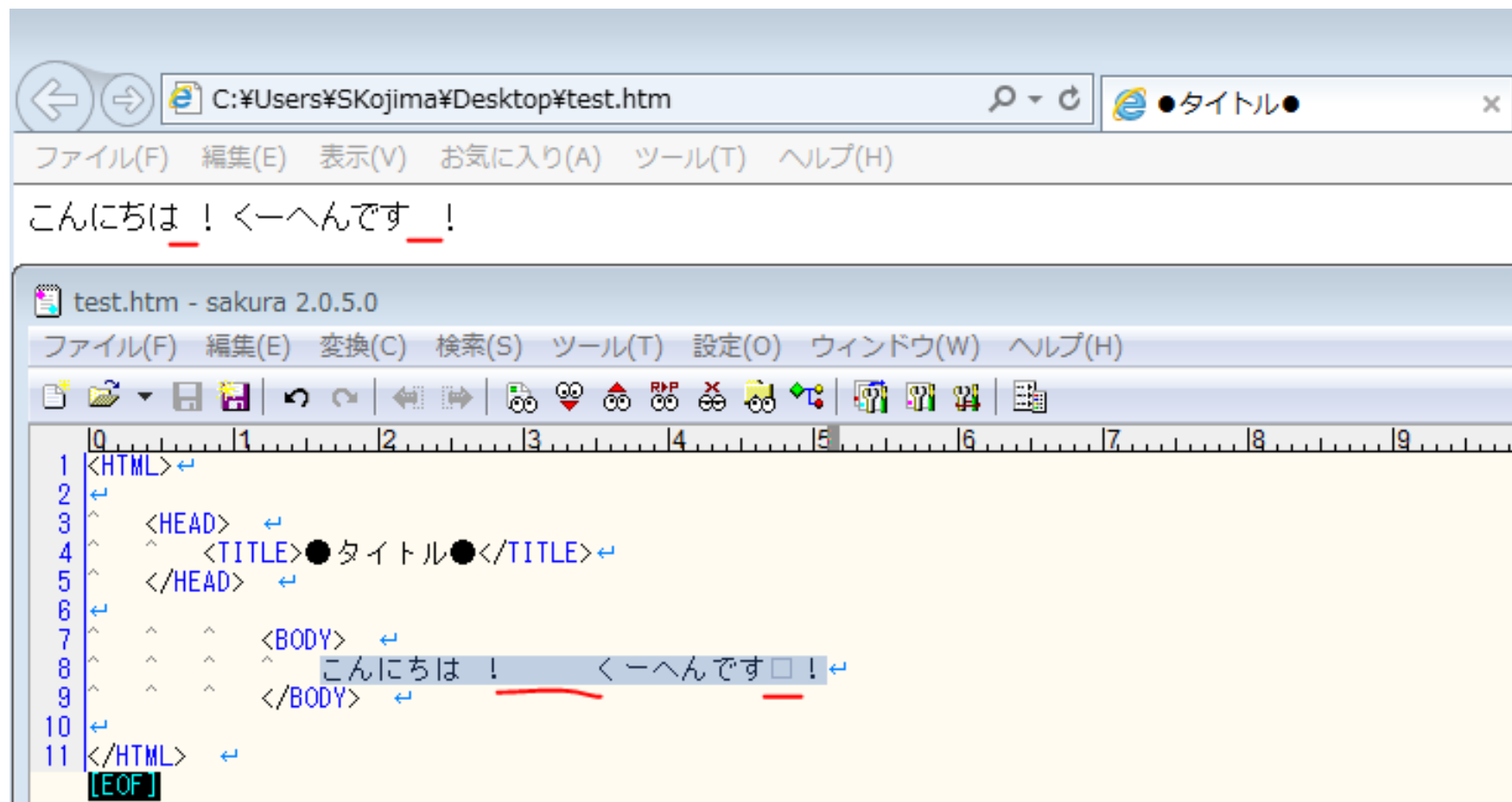
The screenshot shows the Sakura 2.0.5.0 text editor window titled "test.htm - sakura 2.0.5.0". The menu bar includes "ファイル(F)", "編集(E)", "変換(C)", "検索(S)", "ツール(T)", "設定(O)", and "ウィンドウ(W)". The toolbar contains various icons for file operations, editing, and viewing. The editor area displays the following HTML code:

```
1 <HTML> ←
2 ←
3 ^   <HEAD>   ←
4 ^   ^   <TITLE>●タイトル●</TITLE> ←
5 ^   </HEAD> ←
6 ←
7 ^   ^   ^   <BODY>   ←
8 ^   ^   ^   ^   こんにちは ！       くーへんです□！ ←
9 ^   ^   ^   </BODY> ←
10 ←
11 </HTML>  ←
```

<HTML>で始まり </HTML>で終わる

- 取りあえずメモ帳で記述可能
- <タグ>文字列</タグ>で文字を修飾
例: 文字の**大きさ**、文字の**色**等
- 高度になると使用する文字等指定
-
タグを入れないと改行せず表示される
基本ワープロと同じく文字は連続表示
- 使用するブラウザーにより、表示や動作が微妙に違う

半角スペースは無視される罠



Unix OS

- Windows OSが世に出る前は
Unix(ユニックス)という
オペレーティングシステム(OS)が動く
サーバがインターネット上で複数台稼働
しており利用者はそのサーバにTelnetコマンドでサー
バにログインし、sh(シェル)という
対話型のコマンドを使ってメールやNetNews
そしてテキストデータ処理等を行っていた
<https://ja.wikipedia.org/wiki/UNIX>
- Unixのsh(シェル)とは、WindowsのMS-DOSの
「コマンドプロンプト」だと思ってください

正規表現 (せいきひょうげん)

- 正規表現は、**Unixのお陰で一般化**された
- 大量のテキストデータに対し統一した検索ルール(**パターンマッチ**)が必要となり

一括検索: grep(egrep)
一括(検索|置換)、一括削除: sed
一括レコード単位処理: awk(gawk)
C言語、sed、awk、sh(シェル)の
機能を取り入れたプログラム言語: Perl

これらは「正規表現」を使って処理をしている
最近ではWindowsで動くエディタの検索or置換時に「正規表現」の利用が可能



正規表現メモ

<http://www.kt.rim.or.jp/~kbbk/regex/regex.html>

• 2012年8月現在の 「正規表現」使用情報

正規表現メモ

最終更新日 2012年8月24日

正規表現は使い慣れれば便利なものですが、ツールによって使える正規表現演算子(メタキャラクタ)に違いがあったりして戸惑うこともあります。そこで、正規表現を扱うツールの代表的なものをいくつか選び、そこで使われている正規表現演算子をまとめてみました。

正規表現入門者/初心者の方へ

正規表現のチュートリアルが、[perlの正規表現チュートリアル](#)にあります。Perl5.8のドキュメントですが、一般的な入門にも使えると思います。

Table of contents

1. [grep](#)で使える正規表現
2. [egrep](#)で使える正規表現
3. [sed](#)で使える正規表現
4. [awk](#)で使える正規表現
5. [Perl](#)で使える正規表現
6. [Python](#)で使える正規表現
7. [Ruby](#)で使える正規表現
8. [gawk 3.0\(以降\)](#)で使える正規表現
9. [Tcl 8.2.3\(以降\)](#)で使える正規表現
10. [PCRE](#)で使える正規表現
11. [PHP](#)で使える正規表現([mb_ereg](#))
12. [.NET Framework](#)で使える正規表現
13. [Java\(1.4以降\)](#)で使える正規表現
14. [POSIX 1003.2](#)での正規表現について
15. [各正規表現演算子の説明](#)
16. [各エスケープシーケンスの説明](#)
17. [各処理系正規演算子一覧表](#)
18. [ある文字列を含まない正規表現](#)
19. [リンク](#)
20. [用語集](#)
21. [参考文献](#)

Windowsで動く onigsed.exe(20091031版)

GNU sed 4.1.5 の日本語 Windowsへの移植(というほどたいしたことはしていない)です。

cygwinに付属のGNU sedがMBCS対応していないので作業しました。

VC++ 7.1でコンパイルしています。特にDLLは必要ありません。

また、正規表現エンジンとして鬼車(2.5.0)を組み込んでおり、オプション指定によりPerl互換の正規表現を使うことができます。

できるだけオリジナルと同じ振る舞いをするように努力しましたが、どこかに抜けがあるかもしれません(特に改行の扱いに関して)。

漢字コードとしてshiftjis、euc-jp、utf-8 が使えます。使用するときはコマンドラインオプションで

- shiftjis -Wctype=SJIS または --ctype=SJIS (デフォルト)
- euc-jp -Wctype=EUC または --ctype=EUC
- utf-8 -Wctype=UTF8 または --ctype=UTF8

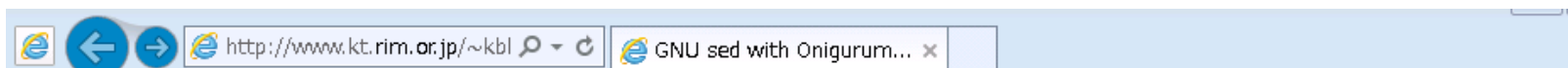
を指定します。マルチバイト処理をオフにしたい場合には

-Wctype=ASCII または --ctype=ASCII

を指定します。

readme ja より抜粋

http://www.kt.rim.or.jp/~kbbk/sed/



GNU sed with Oniguruma (Onigsed)とは?

正規表現エンジンとして、[鬼車](#)を組み込んだ sed です。オプション指定により、POSIX EREや Perl5互換の正規表現を使うことができます。

ダウンロードリンク

- [Onigsed 2009年10月31日バージョン](#)
md5 aae0a510dee4de6acca29cc9c86ca4a5 *onigsed-20091031.zip
sha1 2738e55c05883c334c66dcb264b5ab59202bf8ca onigsed-20091031.zip
- [Onigsed 2009年1月1日バージョン](#)
9728a92c58ef001984a9fed51095535f *onigsed-20090101.zip
0b93c97ba2781a44800e1d888239ac67ec54c62a5f3485196be7d757ebaaf873 *onigsed-20090101.zip
- [鬼車を使っていないバージョン\(2009/10/5\)](#)
md5 3fd753202cb5dbc565eeba29abd049d *sed-mbcs-win32-20091004.zip
sha1 9f4f44f67cb1fa6ad9c95fa3ac232be625b36f6 sed-mbcs-win32-20091004.zip
[\(2009/5/25\)版](#)
949ec827c9383d9ce60c947115db0cdf *sed-mbcs-win32-20090525.zip
8bdc396588ff1e978562e53fbff22e63d5703903 sed-mbcs-win32-20090525.zip

以下、readme.jaより。

・これはなに？

GNU sed 4.1.5 の日本語 Windowsへの移植(というほどたいしたことはしていない)です。cygwinに付属のGNU sedがMBCS対応していないので作業しました。VC++ 7.1でコンパイルしています。特にDLLは必要ありません。また、正規表現エンジンとして鬼車(2.5.0)を組み込んでおり、オプション指定によりPerl5互換の正規表現を使うことができます。できるだけオリジナルと同じ振る舞いをするように努力しましたが、どこかに抜けがあるかもしれません(特に改行の扱いに関して)。

https://github.com/kkos/oniguruma

The screenshot shows the GitHub repository page for **kkos / oniguruma**. The repository is described as a "regular expression library". It has 46 commits, 1 branch, 1 release, and 2 contributors. The latest commit is 45bfe10, made 13 days ago. The repository is currently on the **master** branch. A pull request #3 from Ingramz/patch-1 is merged. The file list includes **cmake**, **doc**, **enc**, **sample**, **win32**, **.gitignore**, **AUTHORS**, and **CMakeLists.txt**. The right sidebar shows links to **Code**, **Issues**, **Pull requests**, **Pulse**, and **Graphs**. The **HTTPS clone URL** is <https://github.com/kkos/oniguruma>. There are buttons for **Clone in Desktop** and **Download ZIP**.

GitHub This repository Search Explore Features Enterprise Pricing Sign up Sign in

kkos / oniguruma Watch 4 Star 15 Fork

regular expression library

46 commits 1 branch 1 release 2 contributors

Branch: master oniguruma / +

kkos Merge pull request #3 from Ingramz/patch-1 Latest commit 45bfe10 13 days ago

cmake	add cmake files.	3 months ago
doc	onig-5.9.2	3 years ago
enc	add onigenc_end_unicode() and entry into end-call-list.	2 years ago
sample	add sample/CMakeLists.txt	3 months ago
win32	onig-5.9.2	3 years ago
.gitignore	add aclocal.m4 into .gitignore	3 months ago
AUTHORS	change mail address.	4 months ago
CMakeLists.txt	add -Wall into FLAGS.	3 months ago

Code

Issues 0

Pull requests 0

Pulse

Graphs

HTTPS clone URL

<https://github.com/kkos/oniguruma>

You can clone with [HTTPS](#) or [Subversion](#).

Clone in Desktop

Download ZIP

Onigsedで → テキスト一括操作 どんなルールがあるのか「メタ文字」

- 行頭 → ^
- 行末 → \$
- 空白行 → ^\$
- 該当データ削除 → /正規表現/d
- onigsed.exe -e “/^\$/d” foo.txt | more

Onigsedで → テキスト一括操作 どんなルールがあるのか「メタ文字」

- 行頭 → ^
- 行末 → \$
- 空白行 → ^\$
- 該当データ置換 → s/正規表現/置換後/g
- onigsed.exe -e “s/^\$/空/g” foo.txt | more

Onigsedで → テキスト一括操作 「g」フラグは2個目以降も置換する

- 該当データ置換 → s/正規表現/置換後/g

```
J:¥>onigsed.exe -e "s/o/p/" test.txt
J:¥fpo.txt      ←最初に見つけた1個だけ置換
J:¥fpo2.txt
J:¥fpo3.txt
J:¥fpo4.txt
J:¥test.txt

J:¥>onigsed.exe -e "s/o/p/g" test.txt
J:¥fpp.txt      ←2個目以降も置換するのが
J:¥fpp2.txt     【 g 】フラグ
J:¥fpp3.txt
J:¥fpp4.txt
J:¥test.txt

J:¥>
```

Onigsedで→テキスト一括操作 【 “ ” 】をパターン内で使えないエラー

- 行全体を置換

`s/^.*&<P>/g`

```
J:¥>onigsed.exe -e "s/^.*<A HREF=&>&<P>/g" test.txt
<A HREF=J:¥foo.txt>J:¥foo.txt<P>
<A HREF=J:¥foo2.txt>J:¥foo2.txt<P>
<A HREF=J:¥foo3.txt>J:¥foo3.txt<P>
<A HREF=J:¥foo4.txt>J:¥foo4.txt<P>
<A HREF=J:¥test.txt>J:¥test.txt<P>
```

↓ ↓ 【 “ ” 】が指定出来ずエラーが出る

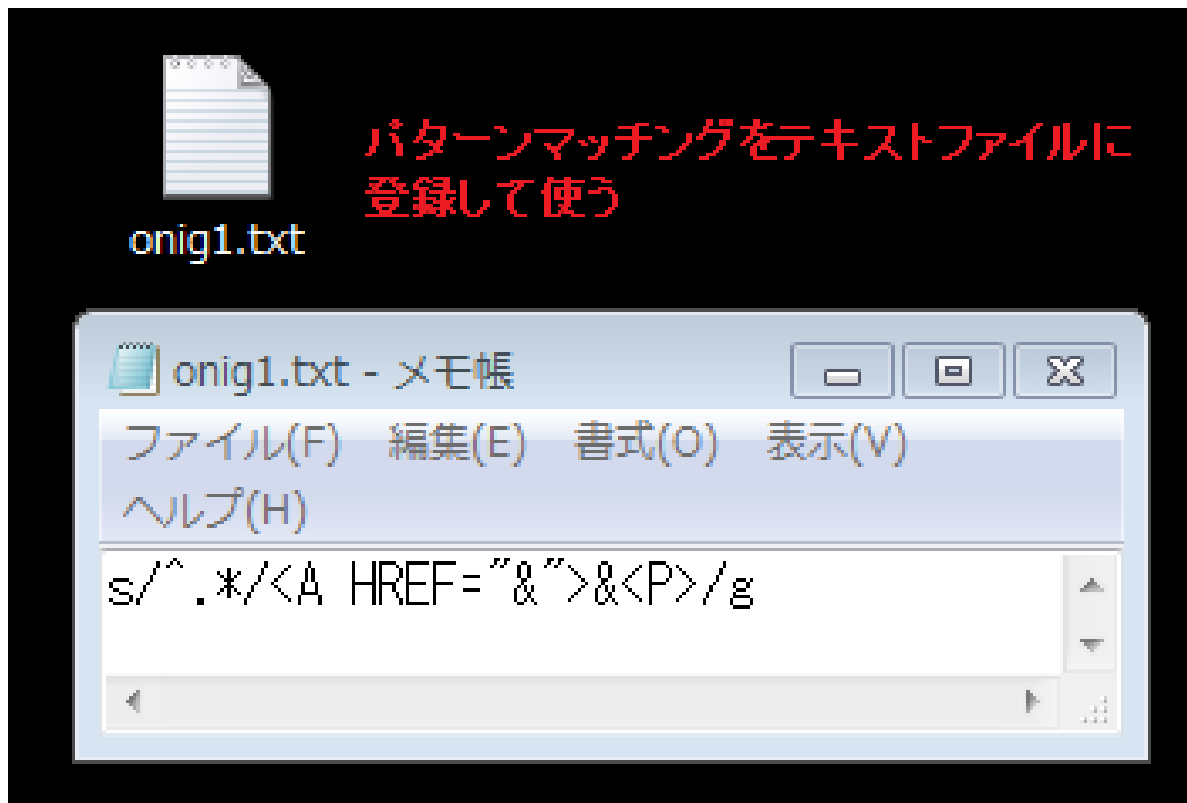
```
J:¥>onigsed.exe -e "s/^.*<A HREF="&">&<P>/g" test.txt
onigsed.exe: -e expression #1, char 14: unterminated `s' command
ファイル名、ディレクトリ名、またはボリューム ラベルの構文が間違っています。

J:¥>
```

Onigsedで→テキスト一括操作 対策：外部ファイルに記述しておく

- 行全体を置換

`s/^.*/&<P>/g`



Onigsedで→テキスト一括操作

onigsed.exe -f 使用ファイル名

- 行全体を置換

`s/^.*/&<P>/g`

```
J:¥>onigsed.exe -f onigl.txt test.txt
<A HREF="J:¥foo.txt">J:¥foo.txt<P>
<A HREF="J:¥foo2.txt">J:¥foo2.txt<P>
<A HREF="J:¥foo3.txt">J:¥foo3.txt<P>
<A HREF="J:¥foo4.txt">J:¥foo4.txt<P>
<A HREF="J:¥test.txt">J:¥test.txt<P>

J:¥>
```

Onigsedで→テキスト一括操作 全行に対して一括処理

- 全行に改行を追加する置換

`s/^.*/&%n/`

```
J:¥>onigsed.exe -e "s/^.*/&%n/" test.txt
J:¥foo.txt

J:¥foo2.txt

J:¥foo3.txt

J:¥foo4.txt

J:¥test.txt

J:¥>
```

メタ文字の【 * 】は 0回以上繰り返された文字列にマッチ

- /Go*gle/

マッチする

0回以上 → Ggle
1回以上 → Gogle

マッチしない

Gpgle
Gopgle

- 出典元
<http://www.rubylife.jp/regexp/repeat/index2.html>

メタ文字の【 + 】は 直前文字に1回以上マッチ

- /Go+gle/

マッチする

Gogle

Google

マッチしない

Ggle

Gopgle

- 出典元
<http://www.rubylife.jp/regexp/repeat/index3.html>

メタ文字の【 . 】は 改行を除く、任意の一文字にマッチ

- /ab.cd/

マッチする

ab^hcd

ab⁴cd

マッチしない

abcd

abppcd

- 出典元
<http://www.rubylife.jp/regexp/repeat/index1.html>

ご質問は？

ニュー◇DB研究所

縦型SQLデータベースモデリング

くーへんの「ライフワーク」 こんな研究やってます

<http://kuhen.jp/ndb/>

2015-02-06秋葉原にて総務省が行った、第一回「異能vation」事業の採択者10名のプレス発表があった同ビルのDMM.make AKIBAで一次選考に入れなかった応募者から約20名選出されて行う「協賛企業マッチング」に参加しました <http://www.inno.go.jp/h26/sponsor.php>

「ライトニングトーク(LT)」と「ポスターセッション」が半分ずつで、私はLTで10分間ダジャレ&ハゲネタを交えて発表したのですが全く受けず → 撃沈しました (事実です)
お時間があれば以下の資料をご覧ください

<https://github.com/flhtc1964/NeuronDB>

以 上