gred.py

概要

grep と sed のあいのこのようなスクリプト。
オプション指定は grep 風。
正規表現指定構文は sed 風だが、マッチ動作は grep 風。

思い出：Windows のコマンドプロンプトで grep が使いたくて作ったツール。ログ解析ソフトのサポートをしてたときにはかなり活躍した。
今は PowerShell でそれなりに高度な正規表現マッチができるので、いい時代になった。

動作要件

Python 3 以上。

- ファイルのエンコードは、デフォルトではシステムのコンソールのエンコード（sys.stdout.encoding）と同じとみなされる。UNIX 系のシステムであれば、環境変数 LANG に依存する。Windows では Shift-JIS とみなされる。エンコードが違うと、ほとんどの場合でファイルオープンの際にエラーになる（Python は自分がデコードできない文字に遭遇すると例外を発生させる）。
- ファイルパスの末尾にコロン（:）で区切って、ファイルのエンコードを明示的に指定することが可能。出力は sys.stdout.encoding になる。

```
> gred.py /regex/ target_file.txt:utf-8
```

- gred は引数として与えられた正規表現パターンを sys.stdin.encoding でデコードするので、上記文字コードの制約をクリアしていれば、日本語のマッチも問題なく行える。

オプション

i: 正規表現で大文字小文字を区別しない。
n: 結果に元のファイルの行番号を表示する。c オプションとは排他。
c: マッチした行をカウントして、その合計結果だけを表示。s/// とは一緒に使えない。また、n オプションとは排他。
v: 指定した正規表現にマッチしない行を表示する。s/// とは一緒に使えない。
H: 表示結果の先頭にファイル名を表示。

grep のように使う

上記すべてのオプションが使用可能（c と n は排他）。
マッチさせたいパターンは、必ず // で囲う必要がある（/ 正規表現 /）。grep のように単純に "pattern" として文字列を指定できない（この辺が sed 風）ので注意。

■ 2008 年 6 月 26 日の 10 時台のログのみ表示：

```
python gred.py "/time=\"?2008-06-26 10:/" example.log
```

- Windows で CPython をインストールした場合、拡張子 ".py" との関連付けが行われるの

で、スクリプトの実行時に Python を明示的に呼び出す必要はない。以下、実行例は明示的に Python を呼び出さない方法で記載する。

```
gred.py "/time=\"?2008-06-26 10:/" example.log
```

- Windows のコマンドプロンプトの場合、正規表現は "" で囲わないと、メタ文字の一部が正しくスクリプトに渡らないので、注意（UNIX シェルの " と同じ）。
- また、ダブルクォーテーション（"）文字を正規表現に含めたい場合は、この例のように \ でエスケープする必要がある。

■ログ中に attack という文字列がある行の行数をカウント（c オプション）：

```
gred.py ic "/attack/" *.log
```

- i オプションは、正規表現マッチで大文字小文字を区別しない。
- ファイル指定にワイルドカードを使用することができる（正確には DOS のワイルドカードではなく、Python の標準モジュール glob を使って、UNIX シェルのグロブ展開を行う）。
- 各ファイルに /attack/ にマッチする行が何行あるか、数字だけが表示される。ファイル名も表示したい場合には、以下のように H オプションも指定する。

```
gred.py icH "/attack/" *.log
```

■ attack という文字列が含まれていない行だけをカウント（v オプション）：

```
gred.py icvH "/attack/" *.log
```

■連続したパターンを指定する（後方参照の使用）：

```
gred.py "/(\d{1,2})\-\1/" example.log
```

- 2 桁までの同じ数字がハイフンで区切られているパターン（9-9、33-33 など）にマッチする。

正規表現内での後方参照は、\number が使える（Python の正規表現の仕様どおり）。

■フィルタ結果に元のファイルの行番号を表示（n オプション）：

```
gred.py n "/port scan/" example.log
```

sed 風の置換コマンド（s///）を使う

s/ 正規表現 / 置換文字列 /
i オプション、n オプション、H オプションが使用可能。
置換処理された文字列が標準出力に出力される。
置換は各行に対してグローバルに行われる。
置換が行われなかった行も、すべて表示される（置換が行われた行のみ表示、という動作はない）。

置換文字列は、Python の文字列エスケープルールにしたがって解釈される（Python は、定義されていないエスケープを発見した場合、エスケープ文字（\）を削除せずに残す）。ただし、例外的にスラッシュのエスケープ（\/）だけはこのスクリプトによって処理され、リテラルのスラッシュ文字（/）に置き換わる。

■ログ中の「tcp/80」という表現を、「http」に置換：

```
gred.py "s/tcp\/80/http/" example.log > replaced_example.log
```

・スラッシュ記号（/）はバックスラッシュ（\）でエスケープする。

■ログ中の「tcp/80」という表現を、「80/tcp」に置換：

```
gred.py "s/(tcp)\/(80)/\2\/\1/" example.log > replaced_example.log
```

・置換文字列内でも、\number で後方参照が使える。re モジュールの re.sub() に渡して処理させているだけなので、置換文字列内だけで有効な \g<number> も使える。