正規表現

データベースと情報検索　第7回

横浜市立大学　坂巻顕太郎
kentaro.sakamaki@gmail.com

ディレクトリの確認

- ¥¥wsl$¥Ubuntu-20.04¥home¥ユーザー名

- cd /mnt/c/Users/ユーザー名/…
  - lsコマンドを使ってアクセスできるディレクトリの確認

ある顧客データのテキストファイル

- sample.txt
  - 青木1020
井上１３２２
上田943
遠藤1013
大沢987
川上９５６
…

- 数値データの問題
  - 半角や全角で入力されている数値データをすべて半角に変換したい

sed コマンド

- sedはStream EDitorの略

```
sed [オプション] スクリプトコマンド ファイル名
```

  - 指定したファイルをコマンドに従って処理する
  - ファイル名を省略した場合は、標準入力からのデータを処理する
  - 出力は標準出力

- sedコマンドの活用
  - パイプとリダイレクトを活用するのが一般的

sed コマンド（つづき）

- テキストファイルに様々な編集を行うコマンド
- yを指定すると文字の置換ができる

```
sed y/０１２３４５６７８９/0123456789/ sample.txt
```

  - 対象文字列の同じ位置にある文字に置換する

- s を指定すると文字列の置換ができる

```
sed s/川上/川田/ sample.txt
```

  - ［置換前］で指定した文字列にマッチした部分を［置換後］に置き換える

区切り文字（デリミタ）

- 区切り文字は伝統的に/ を使うことが多いが、任意の1 文字を利用可能
  - sed -e "s@old@new@g"
  - sコマンドでは、複数マッチした場合は先頭のみが置換されるので、すべてを置換したい場合は、「s/置換前/置換後/g」のように「g」を指定する

- 置換したい文字列に/ が含まれるときなどは他の文字を使う
  - エスケープするために¥をつけて/を使う

grepコマンド

- ファイル中にある文字列（パターン）が含まれる行を表示するコマンド
  - 大文字と小文字は区別される
  - 区別したくないときは-i オプションをつける

- ある文字列を含むテキストファイルを探すこともできる

```
grep 遠藤 *.txt
```

  - grep 遠藤 sample.txtとの違い

正規表現とgrepコマンド

- 「遠」の後に何か書かれていて、その後に0 と書かれている行を知りたい

```
grep 遠藤.*0 sample.txt
```

基本正規表現

記号	意味	使用例
^	行頭	^abc：abcから始まる行
$	行末	abc$：abcで終わる行
¥<	語頭	¥＜abc：abcで始まる単語
¥>	語尾	abc¥>：abcで終わる単語
.	改行以外の任意の文字	―
¥w	アルファベットと数字	―
¥W	アルファベットと数字以外	―
[...]	囲まれている文字のどれか	[abc]：abcのどれか
[^...]	囲まれている文字でない文字	[^abc]：abc以外
[n-n]	指定範囲のどれかの文字	[a-c]：abcのどれか

拡張正規表現

記号	意味	使用例
*	直前文字の0回以上の繰り返し	—
+	直前文字の1回以上の繰り返し	—
{n}	直前文字のn回の繰り返し	[0-9]{3}:3桁の数値
{n,}	直前文字のn回以上の繰り返し	[0-9]{3,}:3桁以上の数値
{,m}	直前文字のm回以下の繰り返し	[0-9]{,5}:5桁以下の数値
{n,m}	直前文字のn回からm回の繰り返し	[0-9]{3,5}＝3桁から5桁の数値
|	または	txt|doc：txtまたはdoc
()	()内をひとまとまりとする	(abc){2}： abcabc

正規表現とgrepコマンド（つづき）

- grep 遠.*0 sample.txt
  - 任意の1 文字（ . ）の0 回以上の繰り返し（ * ）

- grep e..i gpl2.txt
  - e の後に任意の2文字（ .. ）がある
  - スペースなども1 文字に数えられる

- grep 'it¥.' gpl2.txt
  - . という文字にマッチさせたいときは¥. を使う
  - ¥. は特殊な意味があるので、全体をシングルクォートでくくる

¥の役割

- 特殊文字と同じ文字を使いたい場合
  - 「¥」をまえにつける
  - 「¥w」を探したい場合は「¥¥w」

- 正規表現として「?, +, {, |, (,)」を使いたい場合
  - 「¥?」のように「¥」をつける
  - 「-E」オプションをつける

シングルクォートとダブルクォート

- ファイル名やディレクトリ名のスペース
  - cd My Documents などとすると、My とDocuments が別の引数と解釈され、うまくいかない
  - cd "My Documents"とダブルクォートでくくれば、１つの引数とみなされる
  - cd 'My Documents'とシングルクォートでくくるでもよい

- 変数の取り扱い（変数展開が有効かどうか）
  - a="Hello World"; echo "$a" はHello World が表示される
  - a="Hello World"; echo '$a' は$aが表示される

変数

- 変数
  - 代入には「=」を使う
    - =の前後にはスペースを入れない
  - 変数の値を参照するには変数名の前に$をつける
    - 代入の際は$を不要

- 変数名に利用できる文字
  - アルファベット、数字、_のみ

正規表現の例

- grep 命令 gpl2.txt
  - '^You' : Youではじまる
  - '¥.$' : ピリオドでおわる
  - 'in[eg]' : inの後がeまたはg
  - 'in[a-z]' : inの後が小文字
  - 'in[A-Za-z]' :inの後が英文字
  - '19[0-9]' : 19の後が数字
  - 'in[^g]' : inの後がg 以外
  - 'in[^A-Za-z]' : inの後が英字以外

sedコマンド

- 各行ではじめて出てくる19 を20 に置き換えて表示

```
sed s/19/20/ gpl2.txt
```

- 19 を20 に置き換えて表示

```
sed s/19/20/g gpl2.txt
```

- / をスペースに置き換えて表示

```
ls -d /usr/bin/* | sed "s./. .g"
```

正規表現を用いた置換

- 正規表現の使用例

```
sed 's/19[0-9]* /20XX /g' gpl2.txt
sed 's/^/- - -/' gpl2.txt
sed 's/^$/------/' gpl2.txt
```

- ¥(と¥) でくくられた文字列は、前から順に¥1, ¥2, ... として再利用できる

```
sed 's/GNU ¥([A-Z]*¥) ¥([A-Z]*¥)/GNU ¥2 ¥1/g' gpl2.txt
```

正規表現を用いた置換の注意点

- 正規表現はできる限り長くマッチさせようとする

```
sed 's/G.* /gnu /' gpl2.txt
```

  - Gではじまりスペースでおわる文字列にマッチするので、"GNU " にマッチしそうに思うが、実際には"GNU GENERAL PUBLIC " のように、より長い部分にマッチしてしまう

- 以下のようにすれば、上記の問題を避けることができる

```
sed 's/G[a-zA-Z]* /gnu /' gpl2.txt
```

sed コマンド（その他）

- sed /GNU/d gpl2.txt
- sed /^$/d gpl2.txt
  - /正規表現/d でマッチする文字列を含む行を削除
- sed 3d gpl2.txt
  - 3行目を削除
- sed "3iHello World" gpl2.txt
  - 3行目にHello World を挿入
- sed 1,3d gpl2.txt
  - 行の指定は「開始,終了行」のようにつなぐことで範囲指定も可能