ファイル操作

データベースと情報検索　第9回

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

ファイルのダウンロード

- wgetコマンド
  - ノンインタラクティブなダウンローダー

- curlコマンド
  - データを転送するためのコマンド
    - さまざまなプロトコルに対応している
    - アップロードも可能
  - -O オプション：標準出力なしでダウンロードできる

- KaggleやE-Statのデータ
  - APIを使ってアクセスするほうが主流

iris.csvのダウンロード

- Github上にあるzipファイル
  - https://gist.github.com/netj/8836201/archive/6f9306ad21398ea43cba4f7d537619d0e07d5ae3.zip

- 練習
  - wgetコマンドかcurlコマンドを使ってzipファイルをダウンロード

zipファイルを開く

- zipコマンド
  - ファイルをZIP形式で圧縮するコマンド
    - ZIPはWindows環境でポピュラーな圧縮形式

- unzipコマンド
  - ZIP形式で圧縮されているファイルからファイルを取り出す（ZIPファイルを展開する）ためのコマンド

unzipコマンド

- sudo apt install unzip
  - インストールされてない人向け

- unzip [オプション] ZIPファイル [対象ファイル]
  - -d ディレクトリ：指定したディレクトリに展開する
  - CSVファイルだけを展開したい場合、対象ファイルのところに「"*.csv"」を書く
  - 展開先に同名のファイルがある場合、上書きは「y」、すべて上書きは「A」、展開しない場合は「n」、すべて展開しない場合は「N」、ファイル名変更は「r」を入力

練習 : 展開して移動せよ

```
$ unzip 6f9306ad21398ea43cba4f7d537619d0e07d5ae3.zip

$ mv ./8836201-6f9306ad21398ea43cba4f7d537619d0e07d5ae3/iris.csv ./
```

シェルスクリプトの利用

- 毎回コマンドを打つのは面倒
  - 文字コードや改行コードなどを変更するためにiconvコマンドやtrコマンドなどを毎回打つのは面倒

- `./mkcsv.sh file.txt > test.csv`
  - 文字コードや改行コードを変換できるコマンドが書いてあるスクリプト（mkcsv.sh）を利用する
  - .sh: シェルスクリプトの拡張子

シェルスクリプト

- シェル
  - ユーザーからの操作を受け、操作通りに動作した結果を出力するプログラム
  - ユーザーが画面に入力したコマンドを解釈し、カーネルに引き渡す
    - カーネル：CPUやメモリ、ハードディスクなどのハードウェアとソフトウェアを仲介するプログラムで、OSの中核部分
  - Linuxにおける一般的なシェルは、現在はbash
    - 「echo $SHELL」で確認できる

- シェルスクリプト
  - シェルによって解釈・実行される一連の処理を記述したスクリプト
  - 狭義ではUnixシェルで用いられるスクリプト言語

viを使ってshを作成

- vi
  - Linuxで使われるテキストエディタ
  - 簡単な修正などには「vi」か後継の「vim」を使うことが多い

- $ vi ファイル名
  - 新規ファイルの作成

- 新規ファイルの編集と保存
  - "i": インサートモードに変更
  - （好きなように編集）
  - "esc": インサートモードを終了
  - ":w": 保存
  - ":q": viを終了（":wq"とまとめることもできる）
    - ":q!": 変更内容を保存しない場合

シェルスクリプトの例

- Hello Worldを表示するシェルスクリプト
  - $vi hw.sh
  - "i"を入力する
  - $echo "Hello World"
  - escを押す
  - ":wq"を入力してEnterを押す
- 実行権限の所有者への付与
  - chmod 744 hw.sh
- 実行
  - bash ./hw.sh（バッシュでの実行）
  - ./hw.sh（コマンドとして直接実行）

ファイル（フォルダ）のパーミッション

- ユーザ(u)・グループ(g)・全員(a) ごとに読書実行
  - 読む(r), 書ける(w), 実行できる(x) の属性がある
    - Unix の場合、xの属性が立っているファイルが実行ファイル
    - Windowsの場合、exe 拡張子がついたファイルが実行ファイル
  - "ls -l"で"-rw-rw-r- - "と表示されればユーザとグループに属する人は読み書きできるが、他の人は読むことしかできない

- chmodコマンド：パーミッションの変更する
  - chmod a+w ファイル名：みんな書き込める
  - chmod a-w ファイル名 ：みんな書けなくなる

文字コードと改行コードを変更

mkcsv.sh

```
#! /bin/sh
iconv -t SJIS $1 | perl -p -e 's/¥n/¥r¥n/'
```

- 1 行目を#! ではじめると、そのファイルを実行するコマンドを指定できる
  - /bin/sh はUnix 標準のコマンド実行環境
- 2行目 以降に実行したいコマンドを書くことができる
- $1 はコマンドに与えられた1 つ目の引数の意味

- chmod a+x mkcsv.sh
  - mkcsv.shを実行可能にする

- ./mkcsv.sh file.txt > test.csv

練習

- ２つ以上のコマンドを実行するシェルスクリプトを書き、実行してみよ

- 例: a.sh

a.sh

```
#!/bin/sh
echo "Hello world!"
echo "$1"
```

```
chmod a+x a.sh
./a.sh
```

forコマンド

- ある条件に対する繰り返し処理を行う
  - 「1から10まで」のような具体的な数に対する繰り返し
  - 指定したいくつかのファイルに対してある処理を実行
  - など

- forコマンドは主にシェルスクリプトで使用する
  - 複数のファイルに対して同じ処理を行う際にコマンドラインを用いることもある

forコマンドの書式

- 基本的な書式
  - for 変数 in リスト; do コマンド; done
  - for ((式1; 式2; 式3)); do コマンド; done

- 条件
  - for f in *.csv
    - カレントディレクトリにある" *.csv "にあてはまるファイルのすべてにdo以降の処理を施す
  - for i in 1 2 3 4 5
    - do内で変数iの値を参照する場合は$ をつける
    - do echo $i;

forコマンドの例

```
#! /bin/bash
for i in 1 2 3; do
    csvcut -c 1,3,5 a$i.csv > b$i.csv
done
```

```
#! /bin/bash
for ((i = 0; i<100; i=i+1)); do
    csvcut -c 1,3,5 a$i.csv > b$i.csv
done
```

```
#! /bin/bash
rm -rf result
mkdir result
for i in *.csv; do
    csvcut -c 1,3,5 $i > result/$i
done
```

パラメーター展開

- ${変数名〇〇}
  - 〇〇という命令により、変数に格納されている文字列について、文字数の取得、一部の取り出し、変換などができる

- ${変数名::-4}
  - 「末尾の4文字を取り除く」
  - 変数が拡張子付きのファイル名で、そこから3文字の拡張子（.を含め4文字）を取り除くことができる

bashにおけるパラメーター展開 (1)

書式	意味
${変数:-文字列}	変数が未設定または空の場合、指定した文字列を使用する
${変数:=文字列}	変数が未設定または空の場合、指定した文字列を変数にセットする
${変数:?文字列}	変数が未設定または空の場合、エラーとし、指定した文字列を出力する
${変数:+文字列}	変数が未設定または空ではない場合、指定した文字列を変数にセットする
${変数:位置:長さ}	指定した位置から、指定した長さ(最大長)分を取り出す
${変数:位置}	指定した位置以降全てを取り出す
${変数::長さ}	先頭から指定した長さ分を取り出す
${変数::-長さ}	末尾から指定した長さ分を取り出す
${!接頭辞*}	先頭部分が一致する変数名を取得
${!接頭辞@}	先頭部分が一致する変数名を取得
${#変数}	変数の長さ

bashにおけるパラメーター展開 (2)

書式	意味
${変数#パターン}	パターンに前方一致した部分を取り除く（最短一致）
${変数##パターン}	パターンに前方一致した部分を取り除く（最長一致）
${変数%パターン}	パターンに後方一致した部分を取り除く（最短一致）
${変数%%パターン}	パターンに後方一致した部分を取り除く（最長一致）
${変数/パターン/文字列}	指定したパターンを文字列で置き換える
${変数^パターン}	パターンにマッチした最初の文字を、小文字から大文字に変換する
${変数^^パターン}	パターンにマッチした文字全てを、小文字から大文字に変換する
${変数,パターン}	パターンにマッチした最初の文字を、大文字から小文字に変換する
${変数,,パターン}	パターンにマッチした文字全てを、大文字から小文字に変換する

文字列の置換

- 拡張子「.txt」を「.sh」に変更

```
for f in *.txt; do mv $f ${f/.txt/.sh}; done
```

コマンドの実行結果の利用

- 処理したい対象のリストを読み込み、順次処理
  - リストファイルを読み込んで、echoコマンドを実行
  - ファイル名の先頭と最後に「-」を追加

```
for f in `cat ファイル名`; do echo -$f-; done
```

  - コマンドは`` (バッククォート)で括る

- 区切り文字が問題になることがある
  - デフォルトでは空白文字（スペース、タブ、改行文字など）を区切り文字として扱う
    - 文字コードに依存する
  - 特定の区切り文字をIFSで指定する必要がある場合も

練習：シェルスクリプトの作成と実行

- zipファイルを展開する
- 必要に応じて、cdを変更する
- ファイル名を変える
  - ファイル名にある日本語「架電」を半角英数字「Kaden」に変える
- 時間ごとの年齢の平均を求める
  - 各月のデータについて、時間ごとの平均を求める
  - csvkitを使う、awkコマンドを調べて使ってみる、など

.shが実行できない場合

- 「/bin/bash^M: bad interpreter」といったエラー
  - 改行コードが影響している場合があるので、CRLFからLFに置換して実行できるかを確認する

```
sed -i 's/¥r//' *.sh
```

  - その他、文字コードが影響している場合もある