大規模なデータ解析のためのUNIX入門

シェルスクリプト2

23/June/2017

情報管理解析室　西出 浩世

シェルスクリプト応用編：
複数ファイルをまとめて処理する

- 条件違いの同じフォーマットのデータが大量にあるとき
- 一件づつコマンドを実行するのは大変
  - ヒューマンエラーの元にもなる
- SGEのアレイジョブ機能も使えるが、ごく簡単なコマンドには少々面倒

- 複数ファイルに対し繰り返し処理をしてくれるシェルスクリプトの書き方

作業ディレクトリ ~/data/6_script

```
$ cd ~/data/6_script
$ ls
ex2.sh extra
```

- 「SunGridEngine使用方法」の結果を使います
  ~/data/4_sge/results
  に、samファイルが12個あることを確認しましょう

```
$ ls ~/data/4_sge/results/*.sam
```

- 確認できたら、~/data/4_sge/results ディレクトリのシンボリックリンクを同じ名前「result」で作っておきます

```
$ ln –s ~/data/4_sge/results ./results
```

作業ディレクトリ

- ~/data/4_sge/results 内にsamファイルがない場合は下記のようにコピーしてください

```
$ rm -r ~/data/4_sge/results          resultsディレクトリがある場合は消す
$ cp –r /usr/local/data/unix1706/cook/4_sge/results
    ~/data/4_sge/
$ ln –s ~/data/4_sge/results ./results
```

シェルスクリプト：for 文

- 繰り返し処理を実行してくれる「**for文**」
- エディタで新しく下記のスクリプトを記述し、script1.sh として保存
- 実行権を与えてから実行してみる

```
$ emacs script1.sh
```

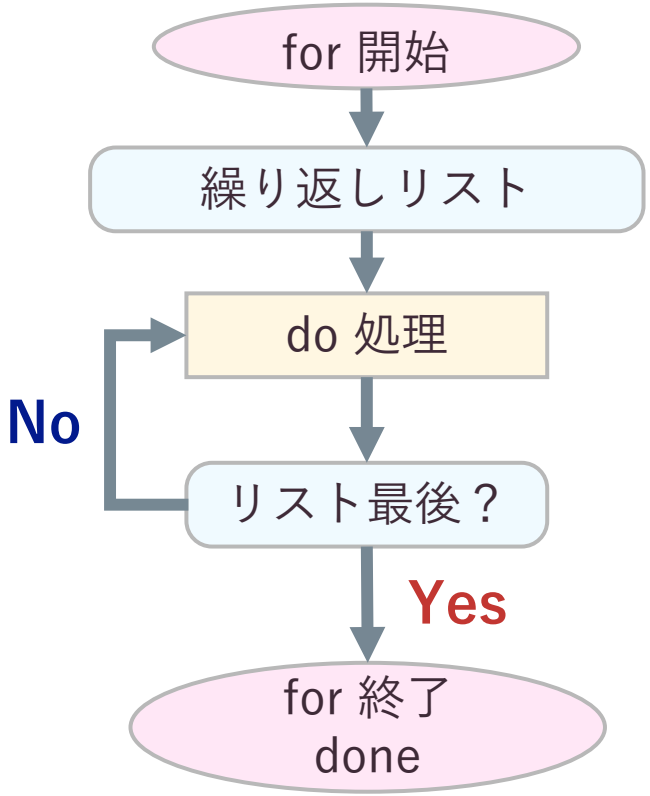
```
#!/bin/sh
for sm in results/*.sam
do
   echo ${sm}
done
```

script1.sh

```
$ chmod +x script1.sh
$ ./script1.sh
results/ecoli.1.sam
results/ecoli.10.sam
results/ecoli.11.sam
results/ecoli.12.sam
```

繰り返し処理：for文

```
for 変数名 in 文字列などのリスト
do
    処理
done
```



- 文字列やファイルのリストに対し、順番にある決まった処理をする
- リストはスペース区切りの文字列挙、配列、数字など
- for 後の変数にリストの値が順番に1つづつ代入され、その後に決まった処理が行われる
- 全てのリストの値が代入され終わったらfor文も終了

for文に使うリストの例

```
for f in ./*
do…
done
```

カレントディレクトリ内にある全てのファイル名をワイルドカード「＊」を使ってリスト（変数 ${f} にファイル名が1つづつ代入される）

```
for i in 1 2 3 4 5 6 7
do…
done
```

1～7までの整数をリスト（変数 ${i} に1～7が順に代入される）

```
for i in {1..10}
do…
done
```

1～10までの整数をリスト（変数 ${i} に1～10が順に代入される）

シェルスクリプト：if 文

- 与えられた条件を判定し、真か偽かで異なる処理をする
- ex2.sh に実行権を与えてから実行してみる
- bowtieの結果ができているかの簡単なチェック

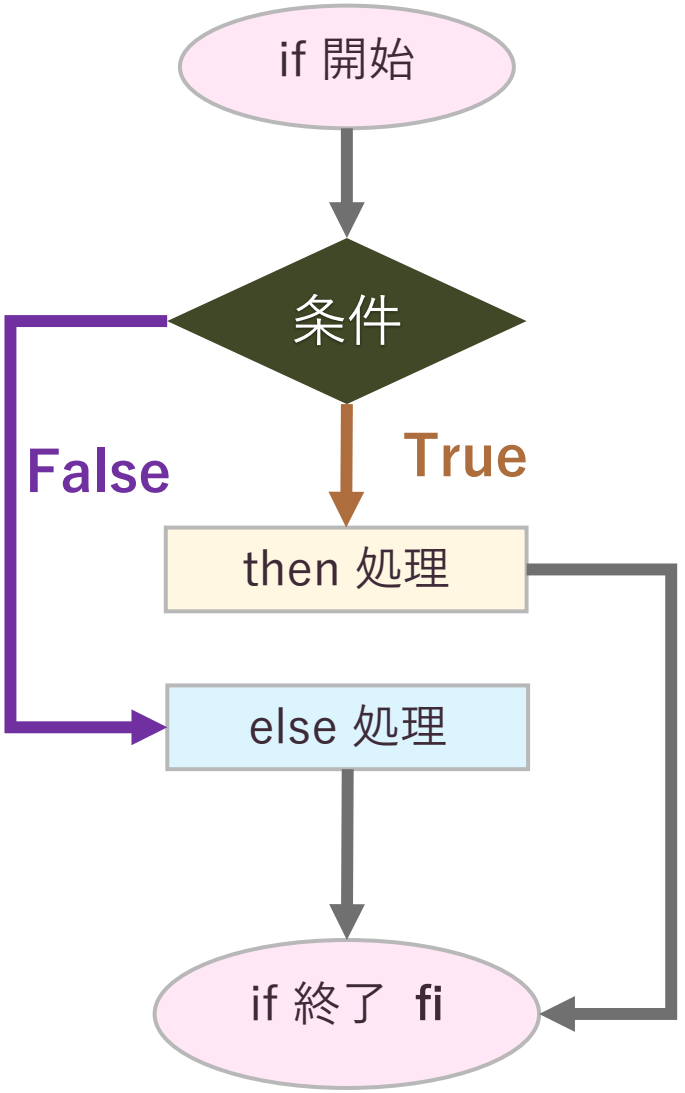
```
$ less ex2.sh
 #!/bin/sh
 if [ -f results/ecoli.10.sam ]
 then
   echo 'ok'
 else
   echo 'not ok'
 fi
```

```
$ chmod +x ex2.sh
$ ./ex2.sh
ok
```

条件分岐：if 文

- ［ ］内の条件が真か偽か？で処理を変える

```
if［条件］
 then
      条件が真だった場合の処理
 else
      偽だった場合の処理
 fi
```

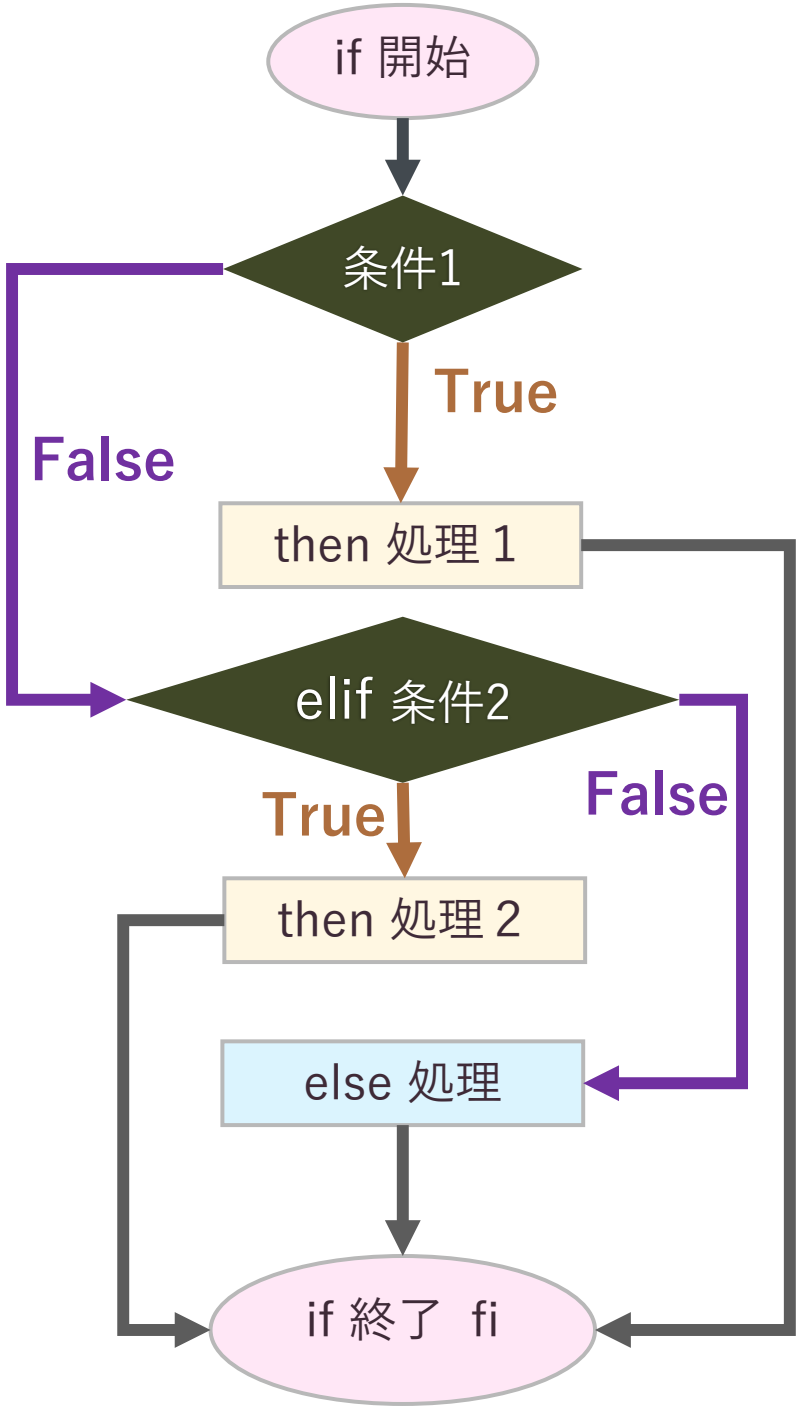


- if → 条件 → then
  　　　　　↘ else
- 「if」は必ず「fi」で終わらねばならない
- 条件を複数設定したい場合は「elif」を使う

条件分岐：if 文

- 複数の条件を設定：**elif**

```
if［条件1］
  then
      条件1が真だった場合の処理
  elif［条件2］
  then
      条件2が真だった場合の処理
  elif［条件3］
  then
      条件3が真だった場合の処理
  else
      全て偽だった場合の処理
fi
```



if : 条件判断の演算子・様々な条件判断

- 条件判断の [ ] は、test コマンドの代替表現
- 下記の演算子一覧は man test で見ることができる

数値比較	
数1 -eq 数2	両辺が等しいと真
数1 -ne 数2	両辺が等しくないと真
数1 -gt 数2	数1 > 数2 の場合に真
数1 -lt 数2	数1 < 数2 の場合に真
数1 -ge 数2	数1 >= 数2 の場合に真
数1 -le 数2	数1 =< 数2 の場合に真

論理結合	
! 条件	条件が偽であれば真
条件1 -a 条件2	条件1, 2 共真であれば真
条件1 -o 条件2	1, 2 どちらかが真であれば真

文字列比較	
-n 文字列	文字列の長さが0でなければ真
! 文字列	文字列の長さが0なら真
文字列1 = 文字列2	両文字列が同じなら真
文字列1 != 文字列2	両文字列が同じでなければ真

ファイルチェック	
-d ファイル名	ディレクトリなら真
-f ファイル名	通常ファイルなら真
-e ファイル名	ファイルが存在すれば真
-L ファイル名	シンボリックリンクなら真
-r ファイル名	読み取り可能ファイルなら真
-w ファイル名	書き込み可能ファイルなら真
-x ファイル名	実行可能ファイルなら真
-s ファイル名	サイズが0より大きければ真
ファイル名1 -nt ファイル名2	1が2より新しければ真
ファイル名1 -ot ファイル名2	1が2より古ければ真

複数のファイル処理

- ~/data/6_script/results/ 内にある12個の .samファイルを .bam ファイルに変換したい
  - アレイジョブを使ってもよいが、今回はfor文を使う

✓ samtools の sam -> bam 変換

```
samtools view -bS example.sam > example.bam
```

- ファイル名は同じにしつつ、拡張子だけを「.bam」としたい

拡張子を変更する

- 「シェルの変数展開」機能を利用する
- 変数展開：変数に保存されいる文字列を置き換える

変数展開の記述	動作
${変数名#パターン}	前方一致での削除
${変数名%パターン}	後方一致での削除
${変数名/置換前文字列/置換後文字列}	文字列置換(最初に一致したもののみ)
${変数名//置換前文字列/置換後文字列}	文字列置換(一致したものすべて)

```
$ a=test.txt
$ echo ${a}
   test.txt
$ b=${a%.txt}
$ echo ${b}
   test
```

削除例

```
$ a=test.txt
$ echo ${a}
   test.txt
$ b=${a/.txt/.cvs}
$ echo ${b}
   test.cvs
```

置換例

拡張子を変更する2

- script1.sh を思い出してみましょう

```
#!/bin/sh
for sm in results/*.sam
do
   echo ${sm}
done
```

script1.sh

- 変数「sm」に、resultsディレクトリ内の.sam ファイルの名前が一つづつ入っています

```
#!/bin/sh
for sm in results/*.sam
do
   bm=${sm/.sam/.bam}
done
```

- としてやれば、変数「bm」に拡張子を.bamに変更したファイル名が入るはず

拡張子を変更する3

- script1.sh をemacs で開き、下記のように改造し「script3.sh」として保存 （C-x, C-w 別名で保存）

```
$ emacs script1.sh
```

```
#!/bin/sh
for sm in results/*.sam
do
   bm=${sm/.sam/.bam}
   echo ${sm} ${bm}
done
```

script3.sh

- 実行権を与えてから実行

```
$ chmod +x script3.sh
$ ./script3.sh
```

演習

- for文を使って results/ 内の全 .samファイルを .bamファイルに変換しましょう
  - samtools の sam -> bam 変換

    ```
    samtools view –b example.sam -o example.bam
    ```
  - script3.sh を改造して作ること
  - 結果のファイル名は、ecoli.1.bam ～ ecoli.12.bam にすること
  - .bam ファイルも results/ 内に保存すること
  - 変数展開を使いましょう
  - SGEオプションも記入し、qsub 用のスクリプトファイルにしましょう

qsub で実行する前に・・・

演習：ポイント

- qsub する前にsamtoolsコマンド全体を echo で出力してみましょう
- 例：

```
echo "samtools view -b example.sam -o example.bam"
```

- echo してみると、変数に入っている文字が正しいかどうかチェックできます。
- テストはqsubではなく ./script3.sh で実施し、echo が上手くいったらecho部分を削除して qsub script3.sh してみてください

ファイル名などに変数を使ったスクリプトでは、
本番実行前に echo で内容を確認しておくとミスを減らすのに有効

演習 途中経過

```
#!/bin/sh

for sm in results/*.sam
do
  bm=${sm/.sam/.bam}
  echo "samtools view -b ${sm} -o ${bm}"
done
```

```
$ ./script3.sh
```

演習 解答

```
#!/bin/sh
#$ -q gitc
#$ -cwd

for sm in results/*.sam
do
  bm=${sm/.sam/.bam}
  samtools view -b ${sm} -o ${bm}
done
```

```
$ qsub script3.sh
```

再びアレイジョブでのbowtie2を考える

- 一連のファイル名に通し番号がついていない場合はどうするか？
- ~/data/6_script/extra には、

  2系統×2条件×3反復 でとった12個のサンプルデータ（fastqファイル）があり、内容に応じた名前がついている。

  （巻末演習問題・実践演習１ 参照）

```
$ cd extra
$ ls
 CFT_MP-1.fastq CFT_MP-2.fastq CFT_MP-3.fastq
 CFT_Ur-1.fastq CFT_Ur-2.fastq CFT_Ur-3.fastq
 MG_MP-1.fastq MG_MP-2.fastq MG_MP-3.fastq
 MG_Ur-1.fastq MG_Ur-2.fastq MG_Ur-3.fastq
```

- これらをSGEのアレイジョブ機能を使ってbowtie2マッピングをするにはどうしたらよいか？

通し番号のないファイル群に対してアレイジョブを使う

- 各ファイルに対し通し番号がついたシンボリックリンクを作る
- 通し番号を作るには？→
  - 通し番号用の変数を一つ用意し、for 文の中で変数をインクリメントする
  - シェルにおけるインクリメント

```
i=$((i+1))
```

または

```
i=$((++i))
```

  - $(())　（二重括弧）でインクリメント演算子が使える
    - 「テキスト処理」最後の演習を思い出してください（ln=ln+1）

通し番号のないファイル群に対してアレイジョブを使う

```
$ less extra.sh
```

```
#!/bin/sh
i=1
for fa in *.fastq
do
  ln –s ${fa} eco_${i}.fq
  i=$((i+1))
done
```

```
$ ./extra.sh
$ ls
```

- ✓ fastq ファイルのシンボリックリンクですが、後に紛らわしくならないように拡張子は「.fq」を使っています。
- ✓ Unixにおいて拡張子はほとんどの場合人の視認用です。