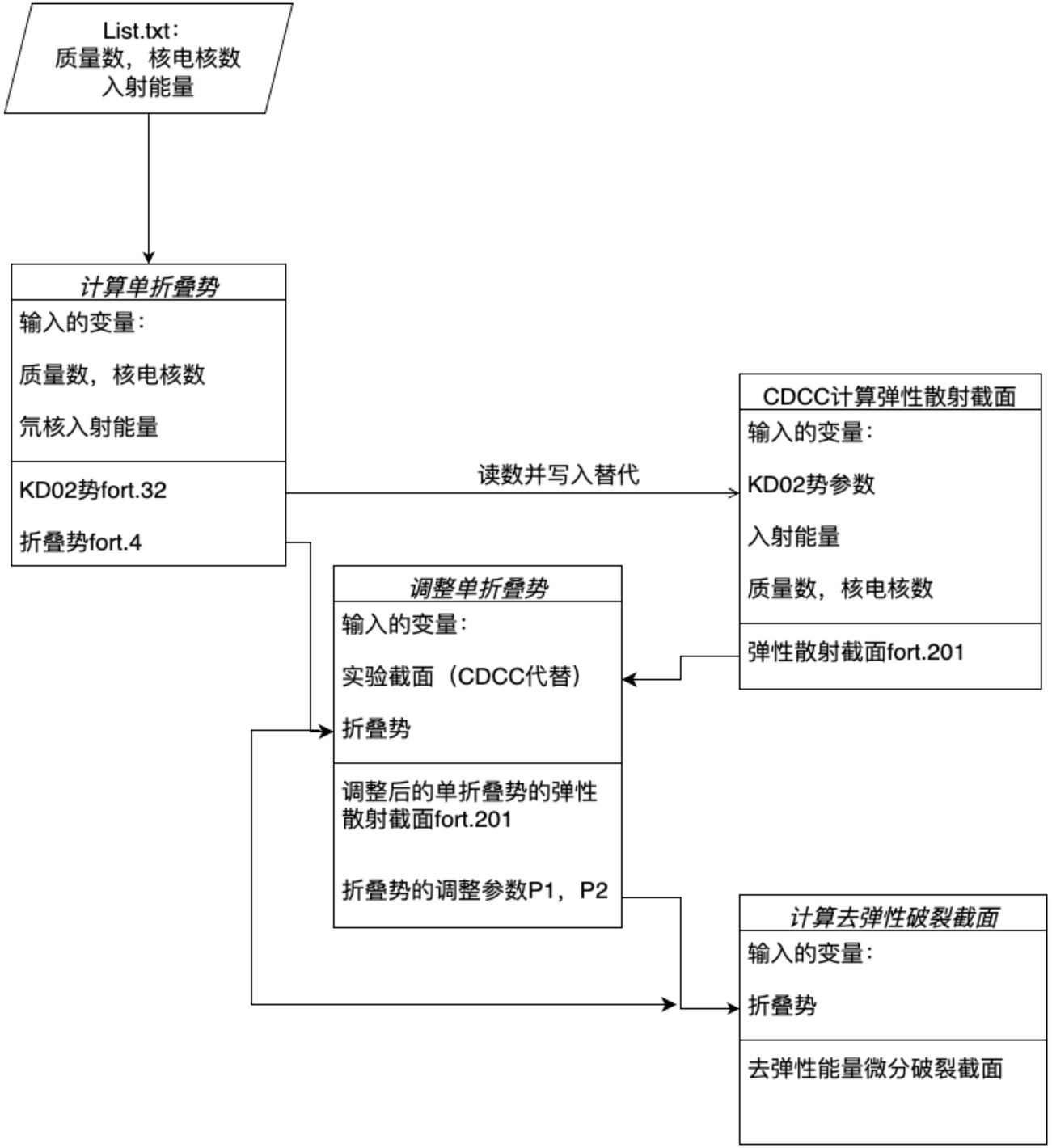
Bash脚本的简单应用与基础命令

大致流程

我希望脚本可以帮我完成以下流程的自动填写，运行，读取等工作，并将结果汇总。



处理的对象

各种Input的形状，如fresco的，计算弹性散射的input范本

```
d+Mo96 Coulomb and Nuclear;
NAMELIST
 &FRESCO hcm=0.1 rmatch=150
         jtmin=0.0 jtmax=50 absend= 0.0010
         thmin=0.00 thmax=180.00 thinc=1.00
         chans=1 smats=2  xstabl=1 wdisk=1
         elab=15.0  nlab=1 /

 &PARTITION namep='d' massp=2.00 zp=1
            namet='Mo96' masst=96.0000 zt=42.0 qval=-0.000 nex=1  /
 &STATES jp=0.0 bandp=1 ep=0.0000 cpot=1 jt=0.0 bandt=1 et=0.0000  /
 &partition /

 &POT kp=1 ap=0.000 at=96.000 rc=1.303  /
 &POT kp=1 type=1 p(1:6)= 94.030  1.150  0.777  2.037  1.327  0.397 /
 &POT kp=1 type=2 p(1:6)= 0.000  0.000  0.000 10.371  1.367  0.815 /
 &POT kp=1 type=3 p(1:6)=  3.557  0.972  1.011 -0.000  0.972  1.011 /

 &pot /
 &overlap /
 &coupling /
```

处理的对象

再看看我们能够得到读到参数的形状，一般在fort.32中

```
An-Cai 06 potentials for a =  96.0
z = 42.0 at  15.000 MeV rc=  1.303
 Vv      rvv     avv     Wv      rw      aw
94.030  1.150  0.777  2.037  1.327  0.397

 Vs      rvs     avs     Ws      rws     aws
 0.000  0.000  0.000 10.371  1.367  0.815

 vso     rvso    avso    wso     rwso    awso
 3.557  0.972  1.011 -0.000  0.972  1.011

```

命令解析

很容易想到一种解决办法，就是读取与替代，读取我们并读入一个数组中我们可以通过正则表示的形式，使用以下命令。

```
values=$(grep -E '^\s*\d+\.\d+\s+\d+\.\d+\s+\d+\.\d+\s+\d+\.\d+\s+\d+\.\d+\s+\d+\.\d+' fort.32)
let j=0;for i in $values ;do v1[$j]=$i;let j++;done
```

Grep是一个常用的读取命令，-E表示使用正则表示，这段表示的\d+\.\d+：表示一个或多个数字，后跟一个小数点，再后跟一个或多个数字。这部分模式用于匹配浮点数。^\s表示开头的多个空格或者没有空格。

将符合改标准的数据读入到values这个变量中，之后再将values中的内容依次读入v1数组中。

命令解析

很容易想到一种解决办法，就是读取与替代，读取我们并读入一个数组中我们可以通过正则表示的形式，使用以下命令。

```
values=$(grep -E '^\s*\d+\.\d+\s+\d+\.\d+\s+\d+\.\d+\s+\d+\.\d+\s+\d+\.\d+\s+\d+\.\d+' fort.32)
let j=0;for i in $values ;do v1[$j]=$i;let j++;done
```

\s+：表示一个或多个空白字符。

$(...)：这是命令替换的语法。它将命令的输出结果替换为该位置的值。在这里，$(grep -E '...' fort.32)将会执行 grep 命令，并将匹配的结果作为字符串返回。

命令解析

Awk命令也是十分常见的读取命令。用于从文件中逐行读取文本并进行处理。它使用指定的脚本来处理输入的每一行数据。

```
#sed -i s/be=-[0-9.]*\+/be=$be/ put.in
awk -v Z2="$Z2" '{sub(/Z2=[^ ]*/, "Z2="Z2)}1' test.in > test.in.tmp && mv test.in.tmp test.in
awk -v E="$E" '{sub(/E=[^ ]*/, "E="E)}1' test.in > test.in.tmp && mv test.in.tmp test.in
awk -v A2="$A2" '{sub(/A2=[^ ]*/, "A2="A2)}1' test.in > test.in.tmp && mv test.in.tmp test.in
```

-v Z2="$Z2": 这是awk的一个选项，用于指定一个变量Z2并将其赋值为Shell环境变量$Z2的值。通过这种方式，将Shell环境变量传递给awk脚本。

'{sub(/Z2=[^]*/, "Z2="Z2)}1': 这是awk的脚本部分。在这个脚本中，使用了sub()函数来替换匹配模式Z2=[^]*的文本，并将其替换为"Z2="后跟Shell环境变量$Z2的值。1表示打印每一行的内容。

命令解析

但是对于一个input中含有多个重名变量，就不可以简单实用匹配替换，还应该指定行位置，进行替换。

```
file="cdc_ni60d_e23.in"
line_number=16
awk -v line="$line_number" -v charge="$Z2" 'NR == line { sub(/charge=[^ ]*/, "charge=" charge) } 1' "$file" > temp.txt
mv temp.txt "$file"
```

与之前不同的是使用-v line=$line_number，指定匹配的范围。即16行的位置上的charge变量，将Z2这个shell内的环境变量写入charge=的后面。

其中sub(/charge=[^]*/，“charge=” charge)是实现这一查找替换的方法。

命令解析

有时我们需要按行依次读取某个文件的数据，进行循环操作，这时候使用read。

```
# 循环处理每个值
while read -r line; do
  read -r name A2 Z2 E <<< "$line"
```

```
done < "list.txt"
```

这段命令是按照list.txt的内容依次读取其中每行的内容输入给line这个shell的环境变量，然后再从line读取其中的name，A2，Z2，E等环境变量。

-r：这是read命令的一个选项，用于禁止对反斜杠字符的特殊处理。通常情况下，反斜杠字符会用作转义字符，用于处理特殊字符，如换行符或引号。但是，使用 -r 选项可以将反斜杠字符视为普通字符，而不进行特殊处理。

命令解析

在流程中我们还会遇到创建文件夹与文件的情况，这时候就需要判别某个文件是否存在，这时候我们需要考虑使用判别

```
#创建yyq势的文件夹，进行smoothie计算
  if [ ! -d "yyq" ]
  then
  # create a new directory to store the result files
  mkdir -p yyq
  fi
  cp ../yyq_smoothie.in  yyq/.
  cp sfm/cm2lab.in yyq/.
  cd yyq
```

-d：表示检查是否存在一个目录”yyq”。而！ 表示取反的意思。

但是值得注意的是这里其实可以直接使用下面的mkdir -p file这一命令来解决，-p表示的是如果目录已存在则不报错，如果目录不存在则创建目录及其上级目录。

命令解析

在shell中可能需要对环境变量进行运算，可以使用let命令（也可以使用echo）

```
let A3=A2+1
```

这里let命令比较简单，echo常用的地方是将某些内容打印到屏幕上，或者写入某个文件中，比如查看shell内设置的环境变量

echo $Z2　　#查看当前环境变量中的Z2的取值

命令解析

在shell中可能需要对环境变量进行运算，可以使用let命令（也可以使用echo）

```
search_file="search.in"

# 设置起始行和结束行
start_line=11
end_line=$(($(wc -l < "$fort201_file") - 2))  # 倒数第二行

# 读取指定行的两列数据，并在第三列增加5%的误差
data=$(awk -v start="$start_line" -v end="$end_line" 'NR >= start && NR <= end { print $1, $2, 0.05 }' "$fort201_file")

# 将数据插入到search.in文件的倒数第3行
temp_file="temp.txt"
line_count=$(wc -l < "$search_file")
insert_line=$((line_count - 2))

head -n "$insert_line" "$search_file" > "$temp_file"
echo "$data" >> "$temp_file"
tail -n +"$insert_line" "$search_file" >> "$temp_file"

# 将临时文件替换为原文件
mv "$temp_file" "$search_file"
```

这一段实现的是将fort.201内的从第11行到倒数第3行的内容读取出来，并在每一行的后面增加0.05表示误差，再将这些内容插入到search.in的倒数第3行中。其中就有echo写入的命令。

命令解析

还有可能遇到我只想要某个文件最后两项的情况

```
last_line=$(tail -n 1 "output-snap")

# 提取最后一行的两个浮点数(即为调整之后的势)
sfmv1=$(echo "$last_line" | awk '{print $1}')
sfmw2=$(echo "$last_line" | awk '{print $2}')
```

```
   88  3.8709E+01   3.87E+01
1.00681E+00 1.15343E+00
   89  3.8709E+01   3.87E+01
1.00682E+00 1.15349E+00
   90  3.8709E+01   3.87E+01
1.00681E+00 1.15348E+00
   91  3.8709E+01   3.87E+01
1.00681E+00 1.15347E+00
   92  3.8709E+01   3.87E+01
1.00681E+00 1.15347E+00
   93  3.8709E+01   3.87E+01
1.00681E+00 1.15347E+00
```

这里可以解释成，将倒数第二行赋予last_line，然后再管道中使用echo将last_line打印到屏幕上，同时使用awk分别读取第一个和第二个打印出来的内容。将其赋予sfmv1和sfmw2。

命令解析

还有可能遇到我只想要某个文件最后两项的情况

```
ecmi=   14.625000000000000
necm=          59
nthcm=         360
nelab=          59
nthlab=         359
inelastic breakup X-sec before transformation is   492.58049925587233
inelastic breakup X-sec after transformation is   491.82758042065439
fort.910 :: double cross sections in CM
fort.911 :: double cross sections in lab
fort.912 :: cross section energy distribution
fort.913 :: cross section angular distribution 1
fort.914 :: cross section angular distribution 2
fort.915 :: double cross sections for fixed angle
fort.916 :: double cross sections for fixed energy
fort.917 :: double cross sections 3-D plot in lab
fort.918 :: double cross sections 3-D plot in cm
fort.919 :: cross sections lx distribution in cm
fort.920 :: cross section for selected angles in cm
fort.921 :: CM angular distribution
```

```
cross=$(awk 'NR==6 {for(i=1;i<=NF;i++) if($i ~ /[0-9]+\.[0-9]+/){print $i;exit}}' cm2lab.out)
cd ..
echo $name  $cross  >> ../name_ac
echo $A2  $cross  >> ../A_ac
```

`'NR==6 {for(i=1;i<=NF;i++) if($i ~ /[0-9]+\.[0-9]+/){print $i;exit}}'`：这是awk的脚本部分。在这个脚本中，使用了条件NR==6来匹配第6行。然后使用for循环遍历该行的每个字段，通过正则表达式/[0-9]+\.[0-9]+/来匹配包含数字和小数点的字段。如果找到匹配的字段，则使用print $i打印该字段的值，并使用exit退出处理，以保证只输出第一个匹配的字段值。