오픈소스SW개론 prj1

가장 먼저 실행 시 보여지는 whoami와 option정보를 init_print()에 구현하고, 출력해주었습니다.

1번 기능

Awk를 통해 구현 하였습니다. 1번의 기능엔 movie_id를 입력 받고, 이를 매개변수 처리하여 func_1에 구현하였습니다. Func_1은 u.item, movie_id를 인자로 갖습니다. movie_id을 awk의 -v 를 통해 awk에서 사용 가능한 변수로 만들어 준 후, awk내에 있는 값과 비교하여 출력해주었습니다. awk에서 print $0는 조건에 만족하는 해당 row를 모두 출력해줍니다. 코드는 다음과 같습니다.

cat $1 | awk -v awk_id=$2 -F '|' '$1==awk_id {print $0}'

```
Enter your choice [ 1-9 ] 1
Please enter 'move id'(1~1682) : 23
23|Taxi Driver (1976)|16-Feb-1996||http://us.imdb.com/M/title-exact?Taxi%20Driver%20(1976)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|1|0|0
```

2번 기능

action장르의 movie를 10개 출력하기 위해, action의 장르가 u.item의 형식에서 어디에 위치하는지 파악하였습니다. 이는 $7이 action의 boolean값을 가지고 있었고, awk를 통해 구현하였습니다. Func_2는 u.item을 인자로 갖습니다. awk에서 Movie번호는 $1에 해당되며, movie이름은 $2에 해당됩니다. 상위 10개를 출력하기 위해 head -n 10을 사용합니다.코드는 다음과 같습니다.

Cat $1 | awk -F '|' '$7==1 {print $1" "$2} | head -n 10

```
Enter your choice [ 1-9 ] 2

Do you want to get the data of 'action' genre movies from 'u.item'? (y/n) : y

2 GoldenEye (1995)
4 Get Shorty (1995)
17 From Dusk Till Dawn (1996)
21 Muppet Treasure Island (1996)
22 Braveheart (1995)
24 Rumble in the Bronx (1995)
27 Bad Boys (1995)
28 Apollo 13 (1995)
29 Batman Forever (1995)
33 Desperado (1995)
```

3번 기능

특정한 movie_id의 rating average를 구하기 위해, u.data에 접근하였고, awk를 통해 rating의 sum 값과 count를 처리해주었고, 맨 마지막 END를 통해 sum,count를 구해주었다. 한번도 rating이 없는 movie_id의 예외 처리를 위해, 한번 더 awk을 수행해주었다. Func_3는 u.data, movie_id를 인자로 갖습니다. U.data에서 movie_id는 awk내에서 $2에 해당합니다. Average rating구하는 코드는 다음과 같습니다.

Cat $1 | awk -v awk_id=$2 -v sum=0 -v cnt=0 '$2==awk_id {print sum,cnt}' |

Awk '$2!=0 { printf("%.6g₩n",$1/$2)}' -> %.6g를 사용해도 상관없는 이유는, 정수부분이 무조건 0~5사이 값을 가지기 때문입니다.

```
Enter your choice [ 1-9 ] 3
Please enter the 'movie id'(1~1682) : 12
average rating of 12: 4.38577
```

4번 기능

4번기능을 구현하며, 이해한 내용을 먼저 얘기하자면, [ ]에 해당하는 조건의 character가 *,+,?,{ } 로 몇 개 표시 되어있는지 알려 줘야 한다. 예를 들어, 전화번호 가운데 4자리를 구별하고 싶으면 각 전화번호 가운데 자리 각각의 형식인 [0-9]를 설정하고, 4자리 이므로 {4}를 뒤에 붙여준다. 다리 말해, [0-9]{4}로 구별 가능하다. IMDb URL는 http[^)]*₩) 로 구별 가능하고, 이를 기술하자면 http로 시작되고,)가 포함되지 않으며 = ^, 마지막은)로 끝나는 부분이다. 이 부분을 빈 문자열 = '' 로 바꿔주기 위해 SED를 사용하였으며, func_4는 u.item을 인자로 갖습니다. 코드는 다음과 같습니다.

Cat $1 | sed -E 's/ http[^)]*₩)/"/' | head -n 10

```
Enter your choice [ 1-9 ] 4

Do you want to delete the 'IMDb URL' from 'u.item'?(y/n) : y

1|Toy Story (1995)|01-Jan-1995|||0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0
2|GoldenEye (1995)|01-Jan-1995|||0|1|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
3|Four Rooms (1995)|01-Jan-1995|||0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
4|Get Shorty (1995)|01-Jan-1995|||0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0
5|Copycat (1995)|01-Jan-1995|||0|0|0|0|0|0|1|0|1|0|0|0|0|0|0|1|0|0
6|Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)|01-Jan-1995|||0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
7|Twelve Monkeys (1995)|01-Jan-1995|||0|0|0|0|0|0|0|0|1|1|0|0|0|0|0|0|1|0|0|0
8|Babe (1995)|01-Jan-1995|||0|0|0|0|0|1|1|0|0|1|0|0|0|0|0|0|0|0|0|0
9|Dead Man Walking (1995)|01-Jan-1995|||0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0
10|Richard III (1995)|22-Jan-1996|||0|0|0|0|0|0|0|0|0|1|1|0|0|0|0|0|0|0|1|1|0
```

5번 기능

5번은 sed의 back-reference를 사용하거나, awk를 통해 구현 할 수 있습니다. 먼저, sed를 하여 | 로 나누어져 있는 각각의 부분을 구별해야 합니다. User_id는 [0-9]+,age는 [0-9]+ or [0-9]{1,2}, gender는 [M],[F], occupation은 [a-zA-Z]+, zip code는 사용 하지 않으니, .*로 처리해주었습니다. 그리고 []로 작성해주어야 하는데, 바로 |를 작성하게 되면, 각각의 조건의 or로 작동되기에, 정상적인 출력이 나오지 않습니다. Func_5는 u.ser을 인자로 갖습니다. 코드는 다음과 같습니다.

조건이 여러 개이므로, -e를 사용하였습니다.

Cat $1 | sed -Ee 's/([0-9]+)[|]([0-9]+)[|][M][|]([a-zA-Z]+)[|](.*)/user ₩1 is $2 years old male $3/

-Ee 's/([0-9]+)[|]([0-9]+)[|][F][|]([a-zA-Z]+)[|](.*)/user ₩1 is ₩2 years old female ₩3/' | head -n 10

Awk로 구현하면 -> cat $1 | awk -F '|' '$3=="M" {printf("user %d is %d years old male %s₩n",$1,$2,$4)} $3=="F" {printf("user %d is %d years old female %s₩n",$1,$2,$4)}' | head -n 10 가 되고, shell에선 string을 = 하나로도 비교 가능하지만, awk에서 =하나만 사용하면 할당의 의미가 되어버린다.

```
Do you want to get the data about users from 'u.user'?(y/n) : y

user 1 is 24 years old male technician
user 2 is 53 years old female other
user 3 is 23 years old male writer
user 4 is 24 years old male technician
user 5 is 33 years old female other
user 6 is 42 years old male executive
user 7 is 57 years old male administrator
user 8 is 36 years old male administrator
user 9 is 29 years old male student
user 10 is 53 years old male lawyer
```

6번 기능

먼저, 영어로 된 month 부분을 숫자로 바꿔줘야 한다. Sed 의 -e를 이용하여 여러 번 바꿔준다. Sed -e 's/Jan/01/' -e 's/Feb/02/' ..... -e 's/Nov/11/' -e 's/Dec/12/' 을 통해 바꿔주고, 해당 결과는 02-01-1995와 같은 형식을 나타내고, 이를 19950102로 바꿔주기 위해 sed를 한번 더 사용한다. 02-01-1995는 ([0-9]+)-([0-9]+)-([0-9]+)로 구별 가능하고, ₩3₩2₩1로 back-reference를 이용하여 바꿔주면 된다. Last 10개를 출력하기 위해 tail -n 10을 사용하였습니다.

Func_6은 u.item을 인자로 갖습니다. 코드는 다음과 같습니다.

cat $1 | sed -e 's/Jan/01/' -e's/Feb/02/' -e 's/Mar/03/' -e 's/Apr/04/' -e 's/May/05/' -e 's/Jun/06/' -e 's/Jul/07/' -e 's/Aug/08/' -e 's/Sep/09/' -e 's/Oct/10/' -e 's/Nov/11/' -e 's/Dec/12/' | sed -E 's/([0-9]+)-([0-9]+)-([0-9]+)/₩3₩2₩1/' | tail -n 10

```
Enter your choice [ 1-9 ] 6

Do you want to Modify the format of 'release data' in 'u.item'?(y/n) : y
1673|Mirage (1995)|19950101||http://us.imdb.com/M/title-exact?Mirage%20(1995)|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
1674|Mamma Roma (1962)|19620101||http://us.imdb.com/M/title-exact?Mamma%20Roma%20(1962)|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0
1675|Sunchaser, The (1996)|19961025||http://us.imdb.com/M/title-exact?Sunchaser,%20The%20(1996)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0
1676|War at Home, The (1996)|19960101||http://us.imdb.com/M/title-exact?War%20at%20Home,%20C%20The%20%281996%29|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1677|Sweet Nothing (1995)|19960920||http://us.imdb.com/M/title-exact?Sweet%20Nothing%20(1995)|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1678|Mat' i syn (1997)|19980206||http://us.imdb.com/M/title-exact?Mat%27+i+syn+(1997)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0
1679|B. Monkey (1998)|19980206||http://us.imdb.com/M/title-exact?B%2E+Monkey+(1998)|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|1|0|0
1680|Sliding Doors (1998)|19980101||http://us.imdb.com/Title?Sliding+Doors+(1998)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0|0
1681|You So Crazy (1994)|19940101||http://us.imdb.com/M/title-exact?You%20So%20Crazy%20(1994)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0
1682|Scream of Stone (Schrei aus Stein) (1991)|19960308||http://us.imdb.com/M/title-exact?Schrei%20aus%20Stein%20(1991)|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
```

7번 기능

u.data에서 user_id를 통해 rate한 movie_id를 가져온다. 코드는 다음과 같다.

Cat $2 | awk -v user_id=$3 'user_id==$1 {print $2}'|sort -n

마지막에 sort를 통해 오름차순으로 정렬해준다. 그리고, 이를 movie라는 변수에 저장하여, 각각의 data를 저장한다. Ex) 4 2 19 234 …

이를 sed를 통해 ' '를 '|'로 바꿔주면 된다. Echo $movie | sed -E 's/ /|/g' 가 코드이다.

그리고 movie에 저장된 movie_id에 해당하는 movie제목을 찾아내기 위해, for문을 사용하여 movie내에 있는 각각의 값을 movie_id로 부여하고 = for movie_id in $movie

Awk를 통해 제목을 찾아내어 movie_id와 name을 출력해준다. 코드는 다음과 같다.

Func_7는 u.item, u.data, movie_id를 인자로 갖는다.

Cat $1 | awk -F '|' -v id=$movid_id 'id==$1 {print $1"|"$2}'

```
Enter your choice [ 1-9 ] 7

Please enter 'move id'(1~1682) : 23

1|7|8|13|14|19|28|32|50|55|56|59|62|70|71|73|79|82|83|88|89|90|91|95|96|98|99|100|102|109|116|124|131|1
32|133|134|143|144|145|151|153|154|155|156|161|162|170|171|172|173|174|175|176|177|181|183|185|188|189|
191|194|195|196|202|203|204|209|211|213|214|215|216|217|219|222|224|227|228|229|230|234|235|238|250|257
|258|269|275|283|294|315|323|357|367|380|381|385|386|387|404|405|408|414|418|419|421|423|427|432|433|44
9|451|463|472|479|483|504|511|512|516|518|522|526|527|528|530|541|546|549|588|597|603|629|642|652|655|6
62|679|694|705|710|713|739|747|780|856|919|961|1004|1005|1006

1|Toy Story (1995)
7|Twelve Monkeys (1995)
8|Babe (1995)
13|Mighty Aphrodite (1995)
14|Postino, Il (1994)
19|Antonia's Line (1995)
28|Apollo 13 (1995)
32|Crumb (1994)
50|Star Wars (1977)
55|Professional, The (1994)
```

8번 기능

먼저, u.user에서 age가 20~29 + occupation이 programmer인 user의 id를 저장해야한다. 이는 awk를 통해 알 수 있으며, 코드는 다음과 같습니다. Age20_29 = $(cat $2 | awk -v occupation="programmer" -F '|"$2>=20 && $2<=29 && $3==occupation {printf("%d₩n",$1)}')

for문을 이용하여 age20_29의 데이터를 하나씩 읽어들이고 = for who in $age20_29

읽어들인 user_id를 통해, u.data에 기록되어 있는 movie_id와 rating을 final_data라는 파일에 저장한다. Cat $1 | awk -v person=$who 'person==$1 {print $2,$3}' >> final_Data

>>를 사용한 이유는 결과를 누적으로 저장해야 하기에 >>를 사용하여 append에 계속 저장해준다.

각각의 movie_id에 따라 average rating을 구하기 위해 1~1682의 movie_id를 모두 확인해주었다. = for num in $(seq 1 1682)

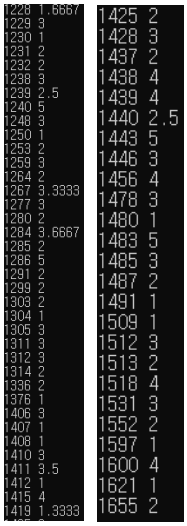
각각의 num에 final_data에 저장된 movie_id에 매칭하여 rating값의 sum을 구해주고 결과를 real_final 파일에 저장한다. Func_8는 u.data, u.user을 인자로 갖습니다. 코드는 다음과 같습니다.

Cat final_data | awk -v n=$sum -v cnt=0 -v sum=0 '$1==n {cnt+=1;sum+=$2} END {print n,sum,cnt}' >> real_final

Real_final에 따로 저장하는 이유는, rating이 할당 되지 않는 movie_id가 존재할 수 있기에, sum/cnt를 출력 할 때 0때문에 오류가 발생한다. Real_final의 데이터를 이용하여 cnt==0인 movie_id를 처리해주고, movie_id와 average rating = sum/cnt를 출력해준다.

Cat real_final | awk '$3 > 0 {printf("%d %.6g₩n",$1,$2/$3)}'

생성된 파일은 rm을 이용하여 제거해준다. Ex. $(rm real_final)



기능 번호 입력 부분 (=enter_choice())

각각의 func_1~8은 enter_choice()에서 호출해주었고, read -p "Enter your choice [1-9] "를 통해 입력 받았습니다. 이후 입력받은 값(=a)는 if [$a -eq 1~8]를 통해 구분해주었습니다.

2,4,5,6,8에서 추가로 (y/n)입력도 read -p " .... (y/n)? " answer로 처리해주었고,

If ["$answer" == "y"]로 string끼리의 비교를 해주어 진행해주었습니다.

Enter_choice함수 내부에 언급했던 부분의 코드를 몇 개 살펴보면,

```
read -p "Enter your choice [ 1-9 ] " a
if [ $a -eq 9 ];then
        echo "Bye!"
        break
fi
```

-enter_choice() 코드 중 option 값 input & exit조건 확인 부분-

```
read -p "Do you want to delete the 'IMDb URL' from 'u.item'?(y/n) : " answer
if [ "$answer" == "y" ]; then
        func_4 $1
else
        echo ""
fi
```

-enter_choice() 코드 중 4번 기능의 (y/n) 입력 후, y입력 시 func_4호출하는 부분-

```
if [ $a -eq 1 ]; then
        read -p "Please enter 'move id'(1~1682) : " movie_id
        func_1 $1 $movie_id
```

-enter_choid() 코드 중 option이 1로 입력 되면, movie_id를 입력 받고 func_1을 호출하는 부분-

```
if [ $# -ne 3 ] || [ $1 != "u.item" ] || [ $2 != "u.data" ] || [ $3 != "u.user" ]; then
        echo "Please type like ₩"./file_name u.item u.data u.user₩""
else
        init_print
        enter_choice $1 $2 $3
fi
```

-초기 입력 예외 처리 부분-