OSS Project #1 Bash Shell Programming Report

인공지능공학과
12223548 백세희

과제 요구사항에 따라 총 9개의 기능을 구현하였으며, 해당 코드와 간단한 설명을 덧붙이고자 한다.

1. 기본 출력 및 파일 설정

```
#! /bin/bash


u_item=$1
u_user=$3
u_data=$2

echo "------------------------------"
echo "User Name: Baek_Sea_Hee"
echo "Student Number : 12223548"
echo "[ MENU ]"
echo "1. Get the data of the movie identified by a specific 'movie id' from 'u.item'"
echo "2. Get the data of action genre movies from 'u.item'"
echo "3. Get the average 'rating' of the movie identified by specific 'movie id' from 'u.data'"
echo "4. Delete the 'IMDb URL' from 'u.item'"
echo "5. Get the data about users from 'u.user'"
echo "6. Modify the format of 'release date' in 'u.item'"
echo "7. Get the data of movies rated by a specific 'user id' from 'u.data'"
echo "8. Get the average 'rating' of movies rated by users with 'age' between 20 and 29 and
'occupation' as 'programmer'"
echo "9. Exit"
echo "------------------------------"
```

쉘 스크립트 파일이 실행 될 때, parameter로 받은 세 개의 파일(u.item, u.data, u.user)을 각각 변수(u_item, u_data, u_user)에 할당하고, 선택할 수 있는 메뉴들을 띄웠다.

2. 사용자 입력 받기

```
while true; do

        echo " "
        read -p "Enter your choice [ 1-9 ] " choice
        case $choice in
```

true일 경우에 한해 계속하여 사용자의 입력을 받을 수 있다. case 구문을 사용해 입력받은 메뉴에 따라 다른 명령을 실행한다.

3. [menu 1] 구현 – case 1

```
                1)
                        echo " "
                        read -p "Please enter the 'movie id' (1~1682): " movie_id
                        echo " "
                        cat "$u_item" | awk -v movie_id="$movie_id" -F "|" '$1==movie_id {print $0}'
                        ;;
```

사용자에게 "movie_id"를 입력받고, "u_item"의 "$1"이 "movie_id"와 일치하면 해당 행을 출력한다.

4. [menu 2] 구현 – case 2

```
2)
        echo " "
        read -p "Do you want to get the data of 'action' genre movies from 'u.item'? (y/n) " y_n
        echo " "
        if [ "$y_n" == 'y' ]; then
                cat "$u_item" | awk -F "|" '$7=="1"{print $1, $2}' | sort -n | head -n 10
        fi
        ;;
```

사용자에게 "y"나 "n" 중 하나를 입력받고, 사용자의 입력이 "y"이라면 다음 코드를 실행한다. "–F "|""를 통해 파이프 기호를 기준으로 "u_item"의 데이터들을 나눈 뒤, 장르-액션과 관련된 값을 가지는 "$7"의 값이 "1"과 같다면 "movie id"와 "movie title"을 출력한다. "sort –n"으로 정렬하고, "head –n 10"으로 상위 10개의 값만 노출시킨다.

5. [menu 3] 구현 – case 3

```
3)
        echo " "
        read -p "Please enter the 'movie id' (1~1682): " movie_id
        sum=0
        counter=0
        average=$(cat "$u_data" | awk -v movie_id="$movie_id" -F "\t" \
                | '$2==movie_id {sum+=$3; counter++} END {printf "%.5f", sum/counter}')
        printf "\naverage rating of $movie_id : $average"
        echo " "
        ;;
```

사용자로부터 "movie_id"를 입력받고, 새로운 변수 "sum", "counter"를 정의한다. "₩t"으로 구분된 "u_data"에서, "$2"가 "movie_id"와 같다면, 해당하는 "$3"의 값을 "sum"에 누적하여 더하고, "counter"의 값을 1만큼 증가시킨다. 시행이 완료된 뒤, 평균값인 "sum/counter"를 "%.5f"를 통해 소숫점 6번째 자리에서 반올림하여 5번째 자리까지 나타낸다.

6. [menu 4] 구현 – case 4

```
4)
        echo " "
        read -p "Do you want to delete the 'IMDb URL' from 'u.item'? (y/n) : " y_n
        echo " "
        if [ "$y_n" == 'y' ]; then
                cat "$u_item" | sed -E 's/http[^|]*//g' | head -n 10
        fi
        ;;
```

사용자에게 "y"나 "n"중 하나를 입력받고, 사용자의 입력이 "y"이라면 다음 코드를 실행한다. "u_item"에서 "http"로 시작하고, 이후 파이프 기호가 나오기 전까지 해당하는 값(IMDb URL)을 모든 데이터에 대해 없애준다. "head –n 10"을 통해 상위 10개의 데이터를 출력한다.

7. [menu 5] 구현 – case 5

```
5)
        echo " "
        read -p "Do you want to get the data about users from 'u.user'? (y/n) : " y_n
        echo " "
        if [ "$y_n" == 'y' ]; then
                cat "$u_user" \
                        | sed -E 's/^([0-9]+)\|([0-9]+)\|([MF]?)\|([a-z]+)\|([0-9]+)$/user \1 is \2 years old \3 \4/g' \
                        | sed -E 's/M/male/g;s/F/female/g' | head -n 10
        fi
        ;;
```

사용자에게 "y"나 "n"중 하나를 입력받고, 사용자의 입력이 "y"이라면 다음 코드를 실행한다. "₩1"은 user id로, "^([0-9]+)"를 통해 하나 이상의 숫자에 해당된다. "₩2"는 age로, "₩1"과 유사하게 하나 이상의 숫자에 해당된다. "₩3"은 gender로, "([MF]?)"를 통해 M이나 F의 값을 0개 또는 1개를 가지게 되는데, 데이터 u_user의

형식에 따라 하나의 값만을 가진다. "₩4"는 occupation으로, "([a-z]+)"를 통해 하나 이상의 소문자 알파벳으로 구성된 값에 해당한다. 이후 형식에 맞춰 출력하고, M을 male, F를 female로 바꾸는 작업을 수행한다. "head –n 10"으로 상위 10개의 값을 출력한다.

8. [menu 6] 구현 – case 6

```
6)
        echo " "
        read -p "Do you want to Modify the format of 'release data' in 'u.item'? (y/n) : " y_n
        echo " "
        if [ "$y_n" == 'y' ]; then
                cat "$u_item" | sed -E 's/Jan/01/g;s/Feb/02/g;s/Mar/03/g;s/Apr/04/g;
                        s/May/05/g;s/Jun/06/g;s/Jul/07/g;s/Aug/08/g;s/Sep/09/g;
                        s/Oct/10/g;s/Nov/11/g;s/Dec/12/g' \
                        | sed -E 's/([0-9]{2})\-([0-9]{2})\-([0-9]{4})/\3\2\1/g' \
                        | sed -n '1673,1682p'
        fi
        ;;
```

사용자에게 "y"나 "n"중 하나를 입력받고, 사용자의 입력이 "y"이라면 다음 코드를 실행한다. u_item에 영문으로 표기되어 있는 month들을 해당 숫자로 교체한다. 이후 sed –E 's/([0-9]{2})₩-([0-9]{2})₩-([0-9]{4})/₩3₩2₩1/g'를 통해 "₩1", "₩2", "₩3"을 추출하고, 이들의 순서를 바꿔 일-월-년 으로 표기되어 있던 release data의 표기 순서를 년-월-일로 바꾼다. "sed –n '1673, 1682p'로 1673부터 1682까지의 movie id를 가진 영화들의 정보만을 출력한다.

9. [menu 7] 구현 - case 7

```
7)
        echo " "
        read -p "Please enter the 'user id' (1~943) : " user_id
        echo " "

        movie_list=$( cat "$u_data" | awk -v user_id="$user_id" -F "\t" '$1==user_id {print $2}'\
                | sort -n | tr "\n" "|" | sed 's/|$//')

        echo "$movie_list"
        echo " "

        m_list=$(echo $movie_list | tr "|" "\n")
        counter=0

        for num in $m_list; do

                counter=$((counter + 1))
                cat "$u_item" | awk -v num=$num -F "|" '$1==num {print $1"|"$2}'
                if [ $counter == 10 ]; then
                        break
                fi
        done

        ;;
```

사용자가 입력한 "user id"와 일치하는 "$1"(user id)의 값을 "u_data"에서 찾고, "$2"(movie id)를 "movie_list"에 추가한다. 이후 "sort –n"으로 오름차순 정렬하며, "₩n"으로 나누어져 있던 정보들을 "tr "₩n" "|""을 통해 파이프 기호로 구분짓는다. 제일 마지막에 표기되는 파이프 기호는 "sed 's/|$//'"로 지워준다. 이렇게 해당 유저가 평가한 영화 목록들이 파이프기호로 구분되어 오름차순 정렬되어 있는 "movie_list"를 출력해준다.

for문의 작동을 위해 "movie_list"의 파이프 기호를 다시 "₩n"으로 바꿔준 다음, 새로운 변수 "m_list"에 저장한다. for문이 "m_list"의 값을 하나씩 읽으며, movie id를 나타내는 "num"의 값을 "u_item"에서 찾아, "{print $1"|"$2}"로 movie id와 movie title을 출력한다. for문이 한 번 완료될 때마다 "counter"가 1씩 증가하며, 총 10번

의 반복을 시행한 후에는 "break"로 for문을 종료한다.

10. [menu 8] 구현 – case 8

```
8)
        read -p "Do you want to get the average 'rating' of movies rated by users with 'age'\
                between 20 and 29 and 'occupation' as 'programmer'? (y/n): " y_n
        echo " "
        if [ $y_n == 'y' ]; then
                user_list=$( cat "$u_user" | awk -F "|" '$4=="programmer" && $2>=20 && $2<=29 {print $1}' | sort -n )

                u_list=$(echo $user_list | tr " " "\n")

                for user in $u_list; do
                        cat "$u_data" | awk -v user=$user -F "\t" '$1==user {print $2"|"$3}' \
                                | sort -n >> movie_rate.txt
                done

                movie_rate="movie_rate.txt"
                declare -A total
                declare -A count

                cat $movie_rate | awk -F "|" '{
                        total[$1] += $2
                        count[$1]++
                }
                END {
                        for (movie_id in total){
                                avg = total[movie_id] / count[movie_id]
                                avg2 = (avg == int(avg) || length(avg) <= 6) ? avg : sprintf("%.5f", avg)
                                printf "%s %s\n", movie_id, avg2
                        }
                }' | sort -n

        fi
        ;;
```

"u_user"에서 occupation에 해당하는 "$4"의 값이 programmer이며, age에 해당하는 "$2"의 값이 20에서 29 사이인 유저의 아이디, "$1"을 "user_list"에 저장하고 오름차순 정렬한다. for문의 작동을 위해 공백으로 나뉘어져 있는 정보들을 "₩n"으로 나누고, "u_list"에 새롭게 저장한다. "u_list"의 값이 하나씩 for문에 전달되며, "u_data"의 "$1"(user id) 값이 "user"에 해당되면, 그 유저가 평가한 movie id "$2"와 평가 점수 rate "$3"을 추출하고, 추출이 완료되면 정렬하여 "movie_rate.txt" 파일에 저장한다. 이 파일을 입력으로 받아 파이프 기호로 나누어준다. "total[$1] += $2", "count[$1] ++"으로 특정 movie id의 rate 값들이 배열 "total"에 누적합 될 것이고, 그 횟수가 배열 "count"에 저장될 것이다. 이후 작업이 끝나면, 존재하는 key값들에 대해 "total"을 "count"로 나누어 평균을 계산하고, "avg2= (avg == int(avg) || length(avg) <= 6) ? avg : sprintf("%.5f", avg)" 는 소숫점이 6자리를 넘어가는 숫자에 대해 반올림하여 5번째 자리까지 나타낸 값을 저장한다. 이후 movie id와 "avg2" 값을 출력하고, 정렬한다.

11. [menu 9] 구현 – case 9, 예외

```
                9)
                        echo " "
                        echo "Exiting the program."
                        exit
                        ;;

                *)

                        echo " "
                        echo "Invalid number! Please enter valid number(1-9)"
        esac
done
```

exit을 통해 while문을 빠져나가게 한다. 1과 9 사이의 값을 입력하지 않은 경우, 예외 메시지를 출력한다.