README

Prj1_12213598_조맑음

목적

Make your script program for handling a input file in **BASH**

Input file

- u.data - user id | movie id | rating | timestamp
- u.user - user id | age | gender | occupation | zip code
- u.item - movie id | movie title | release date | video release date |IMDb URL |Genre…

코드 리뷰

1. 과제 코드에 대한 기본 세팅

```
#!/bin/bash
choice=0
count=0
  #display menu
  echo "--------------------------------------"
  echo "User Name: jomakeum"
  echo "Student Number: 12213598"
  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 user from 'u.user'"
  echo "6. Modify the format of 'release data' in 'u.item'"
  echo "7. Get the data of movies rated by specific 'user id' from 'u.data'"
  echo "8. Get the average 'rating' of movies rated by user with 'age' between 20 and 29 'occupation as 'programmer"
  echo "9. Exit"
  echo "--------------------------------------"
```

- 사용자가 선택할 수 있는 메뉴 `echo` 를 통해 출력한다.
- `choice`, `count`와 같은 코드 실행에 필요한 변수를 우선 초기화한다.

- 학번과 이름은 실제로 지정하였다.

2. 메뉴 선택에 대한 until 루프

```
until [ $choice -eq 9 ]
do
  #case-choice1) get input from user
  read -p "Enter your choice: [1-9] " choice

  case $choice in
```

- 사용자의 선택 입력 값을 `choice` 로 받는다.
- choice가 9가 되면, 프로그램은 종료된다. (9. Exit 선택과 동일)
- choice case 안에서 다음의 1~8번 명령을 수행한다.

3. choice 1번을 선택했을 때

Get the data of the movie identified by a specific 'movie id' from 'u.item'

```
1)
  read -p "Please enter the 'movie id' (1~1682):" movieId
  awk -v movieId="$movieId" -F '|' '$1 == movieId {print}' u.item
  ;; #하나의 case의 끝을 나타낸다
```

- 사용자에게 movie id에 대한 정보를 입력 받은 뒤 `movieId`라는 변수에 저장한다.
- `awk -v` 라는 변수를 정의하고, 초기화 하는 데 사용하는 옵션이다.
- `movieId="$movieId"`로 변수를 선언한다.
- `-F` 을 통해 '|' (pipe) 기준으로 필드를 구분한 뒤, 1번째 필드에 해당하는 값이 movieId와 일치할 경우 해당 필드(행)을 출력한다.
- 필드에 대한 정보는 `u.item`에서부터 가져온다.

---

4. choice 2번을 선택했을 때

Get the data of 'action' genre movies from 'u.item

```
2)
  read -p "Do you want to get the data of 'action' genere moives from 'u.item'? (y/n):" reply
  if [ "$reply" = "y" ]; then
  awk -F '|' '$7 == 1 {print $1, $2; count++} count == 10 {exit}' u.item
  else continue
  fi
  ;;
```

- 2번 선택을 진행할 것인지에 대한 사용자의 입력 값을 `reply` 변수에 저장한다.
- `if [ "$reply" = "y" ]; then` 명령어를 통해, y라고 입력할 경우 아래의 awk를 실행한다.
- `awk` 명령어는 | (pipe) 기준으로 나뉘며, 액션 장르인지 여부를 나타내는 0과 1의 적힌 7번째 필드의 값이 1일 때 (액션 영화 장르가 맞다는 뜻) 1, 2번째 필드를 출력한다.
- count ++는 각 행마다 액션 장르가 있을 경우, +1을 해준다.
- count == 10 (count의 값이 10이 될 경우) 해당 awk 명령을 {exit}를 통해 빠져 나온다.
- 대답이 "y"가 아닐 경우, 다시 메뉴의 선택으로 돌아가기 위해 continue를 추가하였다.

---

5. choice 3번 선택했을 때

Get the average 'rating' of the movie identified by specific 'movie id' from 'u.data'

```
3)
  read -p "Please enter the 'movie id' (1~1682):" movieId
  awk -v movieId="$movieId" -F ' ' '$2 == movieId {sum += $3; count++} END {printf"average rating of %d: %.5f\n", movieId, sum/count}'
  ;;
```

- choice 1번과 마찬가지로 movie id에 대한 사용자의 입력 값을 `movieId`라는 변수에 저장한다.
- `awk -v`를 통해 choice 1번처럼 변수를 선언한 다음 공백을 기준으로 필드를 구분한다.
- 2번째 필드에 해당하는 값이 movieId와 일치한다면, 3번째의 필드의 있는 값을 sum이라는 변수에 누적으로 합해주고, $2 == movieId가 진행될 때마다 하나씩 증가한다.
- 스크립트가 모두 실행되고 마지막으로 print에 해당하는 내용이 출력된다.
- %d에는 movieId에 대한 문자열이 %.6f는 sum/count의 (평균값)의 소수점 6자리에서 반올림한 값이 출력된다.

---

6. choice 4번 선택했을 때

Delete the 'IMDb URL' from 'u.item'

```
4)
  read -p "Do you want to delete the 'IMOb URL' from 'u.item? (y/n):" reply
  if [ "$reply" = "y" ]; then
  cat u.item | sed -E 's/http:[^|]*\)//' | head -n 10
  else continue
```

```
  fi #if문의 종료
  ;;
```

- `u.item`에 대한 내용을 `cat`을 통해 보여준 뒤 sed -E 옵션을 통해 편집한다.
- s를 통해 대체를 나타내는 명령어를 선언한 뒤 http: 로 시작하고, ‘)’로 끝나는 부분을 없앤다.
- `head -n 10`으로 해당 출력값의 상위 10개만 출력한다.

---

7. choice 5번 선택했을 때

Get the data about users from 'u.user’

```
5)
  read -p "Do you want to get the data about users from 'u.user'? (y/n):" reply
  if [ "$reply" = "y" ]; then
  awk -F '|' '{print "user " $1 " is " $2 " years old " $3, $4}' u.user | sed 's/F/Female/g; s/M/Male/g' | head -n 10
  else continue
  fi
  ;;
```

- `awk`의 명령어는 u.user의 데이터를 | (pipe) 기준으로 구분한 뒤 필드 1, 필드 2를 활용한 print문을 출력하도록 한다.
- `sed s`를 통해 F에 해당하는 문자열을 Female로 대체하고, M에 해당하는 문자열은 Male로 바뀐다.
- `sed` 명령어의 /g는 global 옵션으로 입력 텍스트에서 **정규 표현식과 일치하는 모든 항목을 변경**한다.
- 이후 `head -n` 10은 상위 10개만 출력한다.

---

8. choice 6번 선택했을 때

Modify the format of 'release date' in 'u.item’

- Ex) 01-Jan-1995 -> 19950101
- Print only the last 10 lines (movie id: 1673 ~ 1682)

```
6)
  read -p "Do you Modify the format of 'release date' in 'u.item' (y/n):" reply
  if [ "$reply" = "y" ]; then
      while IFS='|' read -r col1 col2 date col4
      do
        new_date=$(date -d "$date" "+%Y%m%d")
        echo "$col1|$col2|$new_date|$col4"
      done < u.item | tail -n 10
  else continue
  fi
  ;;
```

- **IFS - Internal Field Separator**을 통해 문자열을 ‘|’ (pipe)를 기준으로 분할한다.
- 그리고 `-r` 옵션으로 \ (백슬래쉬)를 이스케이프로 처리하지 않게 된다.
  - 여기서 이스케이프는 문자열 내에서 특별한 의미를 가지는 문자(예: 작은따옴표, 큰따옴표, 역슬래시 등)를 표현하거나 특정 문자의 기능을 비활성화하기 위한 메커니즘이다.
  - `-r`은 주로 특수 문자를 그대로 표시하고자 할 때 유용하다.
- `data`의 형식을 새로 바꿔서 `new_data`라는 변수에 저장할 것이다.
  - `date -d` 옵션을 통해 “$data” 변수에 담긴 문자열을 날짜로 해석하도록 한다.
  - 그리고 날짜를 지정된 형식으로 출력하도록 **+** 옵션을 통해 “+%Y%m%d” 문자열을 변경함으로써 원하는 형태로 바꾼다.
- `< u.item`은 해당 while 구문에 사용되는 입력 값이다.
- `| tail -n 10` 를 통해 하위 10개를 출력하도록 하였다.

---

9. choice 7번 선택했을 때

Get the data of movies rated by a specific 'user id' from 'u.data'

```
7)
  read -p "Please enter the 'user id' (1~943): " userId
  movieIds=$(awk -v userId="$userId" '$1 == userId {print $2}' u.data | sort -n | tr '\n' '|') # Sort -n (ascending sort)
  echo "$movieIds"
  echo ""

  IFS="|" read -ra idArray <<< "${movieIds}"
  for id in "${idArray[@]}"; do # @ is every argumetnt
  awk -F '|' -v id="$id" '$1 == id {print $1"|"$2}' u.item
  done | head -n 10
  ;;
```

- 사용자의 입력값을 userId로 저장한다.
- 입력 받을 userId를 변수로 만들어, `u.data`의 1번째 필드와 일치한다면 2번째 필드를 출력하도록 한다.
- 2번째 필드의 내용은 movie id이며 `| sort -n`을 통해 오름차순으로 정렬한 뒤 `| tr`을 통해 모든 개행문자 (\n)을 '|'으로 바꾼다.
- 2번째 필드 내용인 movie id를 `movieId`라는 변수에 저장한다.

- read -ra에서 -a 옵션은 **데이터를 읽어와서 배열**에 저장하라는 지시이다. (문자열은 '|' 로 구분하여 분할된다)
- 분할된 배열은 'idArray'에 저장되며 `<<< "movieIds"`를 통해 **"movieIds"**가 변수의 값을 문자열로 전달하여 데이터 소스 (입력 값)으로 사용된다.
- `<<<`은 Hrer String 문법 중 하나로, **문자열을 명령어의 표준 입력**으로 전달할 때 사용한다.
- 해당 배열의 모든 요소를 `[@]`를 통해 지정하고, id라는 반복변수에 담는다.
- 그리고 각 요소마다 `u.item`이 첫번째 필드의 값과 비교한 뒤 일치한다면 첫번째 필드와 두번째 필드를 '|' 로 구분하여 출력한다.
- 모든 for문이 끝난다면 상위 10개만을 화면에 띄운다.

---

10. choice 8번 선택했을 때

Get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'programmer'

```
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
  if [ "$reply" = "y" ]; then
      awk -F '|' '$4 == "programmer" && $2 >= 20 && $2 <= 29 {print $1}' u.user > user_ids.txt

      while IFS= read -r userId; do
        awk -F ' ' -v id="$userId" '$1 == id {sum+=$3; count++} END {if (count > 0) print id, sum/count}' u.data
      done < user_ids.txt

      rm user_ids.txt
  else continue
  fi
  ;;
```

- user data에서 occupation을 의미하는 4번째 필드의 값이 programmer이고 user의 나이를 뜻하는 2번째 필드가 20살 이상, 29살 이하이면 1번째 필드에 해당하는 user id를 user_ids.txt에 저장하도록 한다.
  - 해당 조건문은 **&&** (and 연산자)로 연결된다.
- `while` 루프는 `user_ids.txt` 파일에서 각 줄을 읽어서 `userId` 변수에 저장하는 역할을 한다.
- `userId`를 `id`라는 변수로 저장한 뒤, `u.data`의 첫번째 필드와 비교한 후 일치한다면, rating에 해당하는 세번째 필드를 누적으로 합하고, 해당 연산이 진행될 때마다 count에 1을 더해준다.
- END { } 블록은 모든 입력 데이터를 처리한 뒤 마무리 작업을 수행하는 데 사용된다.
  - 해당 블록을 통해 count>0 조건이 성립되면 해당 id와 함께 **sum/count (평균)** 값을 출력하도록 한다.
- `rm`을 통해 user_ids.txt 문서를 지운다

---

11. choice 9번 선택했을 때

```
9)
  echo "bye!"
```

```
  ;;

  esac
done
```

- `esac`는 case문의 끝을 나타낸다.
- `done`은 until문의 끝을 의미한다.