proj1_12211723_HongSungMin README

컴퓨터공학과 12211723 홍성민

1. Start & parameter check

```
#! /bin/bash
```

- shebang
  - 스크립트가 기본적으로 bash를 통해 실행되어야 함을 표시

```
if [ $# -ne 3 ]; then
    echo "usage: $0 file1 file2 file3"
    exit 1
fi
```

- positional parameter의 개수 ($#)가 3이 아니라면, 3개의 파일이 모두 주어지지 않은 것이다. 이 경우 usage를 출력하고 에러코드 1으로 프로그램을 종료한다.

```
for f in $*; do
    if [ "$f" == "teams.csv" ]; then
        f_teams="$f"
        continue
    fi
    if [ "$f" == "players.csv" ]; then
        f_players="$f"
        continue
    fi
    if [ "$f" == "matches.csv" ]; then
        f_matches="$f"
        continue
    fi
    echo "Ilegal filename $f"
    exit 1
done
```

- 모든 parameter를 순회하며 각 파일 이름을 찾고, 3개의 올바른 csv 파일이 인자로 들어오지 않았다면 오류 메시지를 출력 후 종료한다.
- 찾은 파일 이름은 변수에 할당하여 이후 파일을 읽을 때 계속 사용한다.
  - 문제에서 요구한 기능은 아니니 굳이 필요는 없었지만, 파일 이름을 하드코딩하면 굳이 parameter를 받을 이유가 없을 것 같아 추가했습니다.

```
for f in "$f_teams" "$f_players" "$f_matches"; do
    if ! [ -f "$f" ]; then
        echo "$f does not exists!"
        exit 1
    fi
done
```

- File 존재 유무 테스트

```
echo "************OSS1 - Project1************"
echo "*        StudentID : 12211723         *"
echo "*        Name : SungMin Hong          *"
echo "***************************************"
echo ""
```

- Who am I
  - 학번과 이름 출력

```
choice=0
while [ "$choice" != "7" ]
do
    echo "[MENU]"
    echo "1. Get the data of Heung-Min Son's Current Club,
          Appearances, Goals, Assists in players.csv"
    echo "2. Get the team data to enter a league position in
          teams.csv"
    echo "3. Get the Top-3 Attendance matches in mateches.csv"
    echo "4. Get the team's league position and team's top
          scorer in teams.csv & players.csv"
    echo "5. Get the modified format of date_GMT in matches.csv"
    echo "6. Get the data of the winning team by the largest
          difference on home stadium in teams.csv & matches.csv"
    echo "7. Exit"
    read -p "Enter your CHOICE (1~7) : " choice

    case "$choice" in
```

- choice 변수를 1로 초기화하고, 그 값이 7(Exit)이 될 때까지 반복한다.
- 반복 루프의 제일 처음에는 메뉴를 출력하고 choice변수에 입력을 받는다. 이후 case를 통해 각 입력값에 맞는 명령을 실행한다.

```
    "1")
        read -p "Do you want to get the Heung-Min Son's data?
        (y/n) : " yes

        if [ "$yes" = "y" ]; then
            awk -F, '$1=="Heung-Min Son" {printf("Team:%s,
          Apperance:%d, Goal:%d, Assist:%d\n", $4, $6, $7, $8)}’
           < "$f_players"
        fi
        ;;
```

1. Get the data of Heung-Min Son's Current Club, Appearances, Goals, Assists in players.csv
   - players.csv를 읽어와 콤마를 구분자로 하여 첫번째 토큰(league_position)이 "Heung-Min Son"이면 Current Club, appearances_overall, goals_overall, assists_overall을 가져와 printf로 출력한다.

```
    "2")
        read -p "What do you want to get the team data of
        league_position[1~20] : " pos
        awk -v p=$pos -F, '$6==p {print $6, $1, $2/($2+$3+$4)}'
          < "$f_teams"
        ;;
```

2. Get the team data to enter a league position in teams.csv
   - 원하는 팀의 리그 순위를 pos변수로 가져오고, awk의 -v 옵션을 이용해 pos 변수를 p라는 이름으로 awk 내에서 쓸 수 있도록 한다.
   - teams.csv를 읽어와 콤마를 구분자로 하여 여섯번째 토큰(full_name)이 사용자 입력값 p와 같다면 이면 league_position, common_name, 를 가져오고, 승률 (wins / (wins+draws+losses))를 계산해 print로 출력한다.

```
    "3")
        read -p "Do you want to know Top-3 attendance data and
                    average attendance? (y/n) : " yes
        if [ "$yes" = "y" ]; then
            echo "***Top-3 Attendance Match***"
            sort -r -n -k 2 -t ',' < "$f_matches" | head -n 3 |
                awk -F, '{printf("\n%s vs %s (%s)\n%d %s\n",
                    $3, $4, $1, $2, $7)}'
        fi
        ;;
```

3. Get the Top-3 Attendance matches in matches.csv
   - 전체 경기들 중 가장 많은 관중수를 기록한 3개의 경기 정보를 출력.
   1. matches.csv를 읽어와 콤마를 구분자로 하여(-t ',') 두번째 토큰(-k 2)을 기준으로 정렬하고, -n옵션을 사용하여 각각을 숫자로써 비교한다(-n이 없으면 알파벳 순으로 비교하여, 9980이 81332보다 더 큰 값으로 인식하게 된다) .
   2. head로 상위 3개만 가져온다.
   3. awk를 통해 경기당 두 팀의 이름, 날짜, 관중수와 경기장 위치를 출력한다.

```
    "4")
        read -p "Do you want to get each team's ranking and the
                    highest-scoring player? (y/n) : " yes
        if [ "$yes" = "y" ]; then
            OLDIFS=$IFS
            IFS=,
            for team in $(sort -n -k 6 -t ',' < "$f_teams" |
     awk -F, '$1!="common_name" {printf("%s/%s,", $6, $1)}'); do
                echo ""
                echo "$team" | tr '/' ' '
                awk -v t=$(echo "$team" | cut -d '/' -f2) -F,
     '$4==t {printf("%s/%s\n", $1, $7)}' < "$f_players" |
     LC_CTYPE=C sort -r -n -k 2 -t '/' | head -n 1 | tr '/' ' '
            done
            IFS=$OLDIFS
        fi
        ;;
```

4. Get the team's league position and team's top scorer in teams.csv & players.csv
   - 리그 순위 순서대로 각 팀별 최고 득점 선수를 가져와 출력한다.

   1. for loop에서 콤마를 구분자로 동작하기 위해 IFS 변수의 값을 콤마로 바꾼다. 이후 작업이 끝난 후 다시 원래 값으로 되돌려놓기 위해 원래 값을 OLDIFS에 저장한다.
   2. for로 순회할 데이터를 생성한다
      1. teams.csv를 읽어와 sort로 리그 순위(6번째 토큰)을 기준으로 정렬한다.
      2. awk로 파이프라이닝한 후, 1번 토큰이 "common_name"이 아닌 경우(csv파일의 맨 위 목차 부분) 리그 순위(6번)과 팀 이름(1번)을 중간에 /를 넣는다. 이렇게 하면 결과값은 "1/Manchester City"와 같이 되는데, 구분자로 공백이 아닌 /를 넣는 이유는 이후의 명령에서 위의 결과값을 잘라 팀 이름을 얻기 위해서다.
      3. 결과적으로는 "1/Manchester City", "2/Liverpool"과 같은 데이터가 순서대로 team 변수에 들어가 for loop가 돌게 된다.
   3. 우선 처음에는 팀의 순위와 이름을 출력해야 한다. echo "$team" | tr '/' ' ' 명령으로 team변수의 "/"문자를 공백으로 바꾸어 출력한다.
      ("1/Manchester City" —> "1 Manchester City")
   4. 이후 해당 팀에서 가장 득점 수가 많은 선수를 찾는다.
      1. 팀의 이름을 가져오기 위해, team변수의 값을 cut에 넘기되, "/"를 구분자로 하여 2번째 토큰을 구한다. 만약 공백을 기준으로 잘라 두번째 토큰을 구할 경우, "1 Manchester City"에서는 "Manchester"만 구해지게 된다.
      이렇게 구한 값을 awk의 -v 옵션을 통해 변수 t로써 awk 안에서 사용한다.
      2. players.csv 값을 읽어와 4번째 토큰(소속팀)이 변수 t와 같다면 이름과 골 수(1, 7번 토큰)을 가져와 "/"를 사이에 두고 출력한다.
      이때도 "/"를 삽입하는 것은 이후의 sort 명령에서 구분자를 "/"로 하기 위함이다.
      3. 2번의 값을 sort를 이용하여 정렬하는데, 이때 구분자를 "/"로 한다.
      만약 공백을 구분자로 하게 되면, 선수의 이름 사이에 공백이 있는 경우 문제가 생길 수 있기 때문이다.
      또한, 선수 이름 내에 있는 특수문자에 의해 sort명령에서 오류가 생길 수 있으므로 LC_CTYPE=C 설정을 붙여 실행한다.(mac bash에서는 해당 문제로 인하여 LC_CTYPE=C 가 필요한데, 리눅스 가상머신의 경우는 어떠한지 테스트 해보지는 못했습니다)
      4. 이후 정렬된 값들의 제일 첫번째 값을 가져와(head -n 1), "/"를 공백으로 바꾸어(tr '/' ' ') 출력한다.
   5. 변수 IFS에 원래 값을 다시 저장한다.

```
    "5")
        read -p "Do you want to modify the format of date? (y/n)
                     : " yes
        if [ "$yes" = "y" ]; then
            awk -F, '$1!="date_GMT" {print $1}' < "$f_matches" |
            head -n 10 |
            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 -e 's/([0-9]{2}) ([0-9]{2}) ([0-9]{4}) \-
([0-9]+\:[0-9]{2}(am|pm))/\3\/\1\/\2 \4/'
        fi
        ;;
```

5. Get the modified format of date_GMT in matches.csv
   - 날짜 데이터를 변환하여 출력한다.
   1. 우선 awk로 matches.csv파일을 읽어와 첫번째 토큰만을 출력한다(첫번째 토큰이 "date_GMT"인 경우 파일 맨 위의 목차 부분이므로 무시한다.
   2. 이중 제일 위의 10개 값만 가져온다 (head -n 10)
   3. sed를 이용해 월 이름을 숫자로 바꾼다. substitution 명령을 이용해 각 월에 해당하는 영어 이름을 숫자로 바꾸고, -e 옵션을 이용해 s명령을 여러 개 사용한다.
   4. 이후 다시 sed를 이용해 포멧을 바꾸는데, "mm dd yyyy - h:mm(am|pm)"의 포멧으로 되어 있는 원본을 "yyyy/mm/dd h:mm(am|pm)"으로 바꾸기 위해 정규 표현식으로 각각을 그루핑해 순서를 바꾼다.
      1. ([0-9]{2}) <— 숫자 2개(월)
      2. ([0-9]{2}) <— 숫자 2개(일)
      3. ([0-9]{4}) <— 숫자 4개(연도)
      4. ([0-9]+\:[0-9]{2}(am|pm)) <— 숫자 1개 이상(시간)과 이스케이프된 콜론(:) 뒤의 숫자 2개(분), 마지막으로 am 또는 pm 문자열
      5. 이렇게 4개의 그룹을 만들어 "(3번)/(1번)/(2번) (4번)"의 포멧으로 재배열한다.

```
    "6")
        awk -F, '$1!="common_name" {printf("%s) %s\n", NR-1,
                 $1)}' < "$f_teams"
        read -p "Enter your team number : " team
        echo ""
        selected=$(awk -v t="$(( $team+1 ))" -F, 'NR==t
                                {print $1}' < "$f_teams")
        for i in $(awk -v s="$selected" -F, '$3==s
                    {print $5-$6}'  < "$f_matches"); do
            if [ ${max-$i} -le $i ]; then
                max=$i
            fi
        done
        awk -v m="$max" -v s="$selected" -F, '$3==s && $5-$6==m
         {printf("%s\n%s %d vs %d %s\n\n", $1, $3, $5, $6, $4)}'
         < "$f_matches"
        unset max
        ;;
```

6. Get the data of the winning team by the largest difference on home stadium in teams.csv & matches.csv
   - 선택한 팀이 홈 구장에서 가장 큰 점수차로 이긴 경기들을 출력한다.
   1. 우선 선택을 위해 각 팀의 번호와 이름을 순서대로 출력한다.
      - teams.csv를 읽어와 1번째 토큰이 "common_name"(목차)이 아니라면 해당 팀의 번호(NR-1)와 이름을 출력한다.
   2. 원하는 팀의 번호를 선택받고, awk를 이용해 거꾸로 번호에 맞는 팀 이름을 가져와 selected변수에 해당 이름을 저장한다.
      - -v옵션으로 team변수(사용자가 입력한 번호) + 1값을 변수 t로써 사용하고, NR의 값이 t와 같을 때만 첫번째 토큰(팀 이름)을 가져온다.
   3. 해당 팀의 모든 경기를 확인하기 위해, awk를 통해 matches.csv에서 3번째 토큰(홈 팀 이름)이 selected변수값과 같은 모든 경기에 대해 점수 차를 모두 계산하여, 각각을 for문으로 순회한다.
   4. 최댓값을 구하기 위해 max변수를 사용하는데, max값이 i보다 작거나 같으면 max값을 i값으로 바꾼다. 이때 max값이 unset인 경우(for문의 제일 초기)인 경우 조건식에서 max대신 i값을 쓰는데, 이 경우를 위해 -lt가 아닌 -le를 사용한다. -lt를 사용하는 경우 for문의 초기 경우에 [$I -lt $I]이 되어 조건식이 항상 false가 되므로 max변수의 값은 끝까지 unset이 되기 때문이다.
   5. 골 차이의 최댓값을 max변수에 저장한 이후, awk를 통해 팀 이름과 골 차이가 맞는 경기 데이터만을 골라 출력한다.

```
    "7")
        echo "Bye!"
        exit 0
        ;;
```

7. Exit
    - Bye!를 출력하고 프로그램을 종료한다.

```
    esac
    echo ""
done
```

- 각 case에 맞는 처리가 다 되면 한 줄을 띄운 후, 다시 while문 내의 맨 위로 돌아가 메뉴 출력부터 다시 반복한다.