OSS Project 1

학번 : 12211607 이름 : 박상경

1. 전체적인 프로그램의 구상

우선 csv 파일 3개를 매개변수로 같이 받아와 실행하는 방식이므로, 코드 시작 부분에 매개변수에 대한 예외 처리를 해주는 게 좋다고 생각했다. 이후 예외가 발생하지 않았다면, 학번과 이름이 출력되며 1~7까지 입력받아 진행되는 프로그램이 실행이 된다.

프로그램의 진행은 7이 입력되기 전까지는 계속 반복되어야 하기 때문에 until문을 이용하여 매개변수에 7이 입력되지 않는다면 반복 실행이 가능하게 구현하고, case문을 활용하여 입력받은 수에 따라 실행되는 방식이 구현하기 편할 것 같아 이와 같이 계획하게 되었다.

1. 프로그램의 세부 설명
2. 첫 예외 처리

csv 파일 3개를 매개변수로 받아와야 하기 때문에 if문을 하나 사용하여 매개변수의 개수가 3개가 아니라면 지정된 에러 메시지가 뜨게 구성하였다. 또한, 매개변수로 받아서 코드 내에서 활용하려면 csv가 입력되는 순서도 중요했기 때문에 else if를 하나 추가하여 매개변수 $1 $2 $3 각각이 teams.csv, players.csv, matches.csv가 아니라면 역시 지정된 에러 메시지를 출력하도록 하였다. 당연히 에러 메시지가 출력된 후에는 프로젝트 파일을 종료할 수 있도록 exit 1을 걸어두었다.

1. 프로젝트의 큰 틀

앞서 언급했듯이, 7이 입력되기 전까지는 계속 반복되어야 하기 때문에 until문을 사용하여 choice라는 변수가 7이 될 때까지 반복한다는 조건문을 만들어 주었고, 그 전에 최초로 프로젝트 파일을 실행하였을 때 학번과 이름이 출력되도록 until문 바깥에 기본 정보들을 작성해 두었다.

until문 내부에는 1번부터 7번까지의 선택 보기들이 나열되어 있고, read 명령어의 p 옵션을 활용하여 입력받은 값을 choice 변수에 대입하였다. 그리고 choice 변수의 값에 따라 지정된 기능을 할 수 있도록 case문을 활용하여 7가지의 상황들을 만들어 놓았다.

yes / no 의 선택지가 있는 곳과 특정 값을 입력받아야 하는 경우 모두 read -p를 활용하여 변수에 값을 지정했다. y/n 중 선택해야 하는 상황일 때 n을 입력했을 경우 “retry!” 메시지가 뜬 후 다시 처음으로 돌아가는 방식으로 no에 대한 상황 처리를 해주었고, 위와 같은 방식으로 1~20의 범위 중 하나를 선택해야 하는 상황일 때도 범위를 벗어나게 된다면 예외 처리를 해주었다.

1. Case 1

1번 기능은 손흥민 선수의 기록을 열람할 것인지 물어보고, y가 입력되었다면 손흥민 선수의 팀, 출전 경기 수, 골 수, 어시스트 수를 출력하는 것이다. 마찬가지로 read -p를 활용하여 yn 변수에 저장하고, y가 입력되었다면 players.csv 파일에서 awk를 활용하여 필요한 부분만 가져와 출력하였다. 선수 이름 중 Heung이라는 문자열이 있는 경우는 없었기에 $1~”Heung”으로 처리하였다.

1. Case 2

2번 기능은 팀 순위를 입력하면 해당 팀의 순위와 이름, 승률을 출력하는 것이다. 1번과 마찬가지로 read -p를 이용하여 수를 받고, teams.csv 파일에서 입력한 순위와 파일의 $6을 비교하여 일치한다면 해당 행에서 승, 무, 패 기록을 가져와 승률을 계산하여 출력했다. 여기서 주의할 점은, bash 변수를 awk에서 사용할 땐 v 옵션을 이용하여 한번 더 바꿔주는 작업이 필요하다는 것이다.

1. Case 3

3번 기능은 관중 수가 가장 많았던 경기 3위까지 경기 대진과 일시, 관중 수, 장소를 출력하는 것이다. matches.csv 파일에서 관중 수가 있는 부분인 $2 부분을 기준으로 sort 명령어를 사용했다. 여기서 추가로 사용한 건, t 옵션으로 구분자를 지정해줄 수 있다는 것이다. -t, 이와 같이 사용하면 ,를 기준으로 행이 나누어지게 되어 -k 2를 해주면 쉽게 정렬이 가능하다. 하지만 이렇게 해도 문제가 발생하는데, 9000이 80000보다 앞자리의 수가 크다는 이유로 우선순위를 가진다는 것이다. 이를 해결하기 위해 옵션 n을 같이 써준다면 문자열이 아니라 수로 인식하여 정렬하게 되어 수가 큰 순서대로 정렬이 가능하게 된다.

1. Case 4

4번 기능은 팀 순위 순서대로 각 팀에서 가장 많은 득점을 기록한 선수와 득점 수를 출력하는 문제다. 여기서 가장 많이 헤매었는데, 그 이유는 result로 공백이 포함된 문자열을 얻어오게 되면 awk 내에서 bash 변수를 불러올 때 공백 때문에 에러가 생기게 된다는 것이었다. 한참을 고민하고 인터넷을 찾아보던 끝에 v 옵션에서 a=$result가 아닌 a=”$result”로 double quotes로 묶어주면 해결이 된다는 것을 알게 되었다. for문을 이용하여 1~20위 팀명을 result에 저장한 후, teams.csv에서 팀 순위와 팀명을 출력한다. 그 후에 players.csv에서 득점 순서대로 정렬한 후, awk를 통해 선수가 속한 팀과 result를 비교하여 일치한 선수들을 골라낸 후, 가장 위에 있는 값만 출력하여 문제를 해결했다.

1. Case 5

5번 기능은 sed를 활용하는 문제였다. 가장 상단에 있는 10개의 데이터를 head -n 10을 통해 활용하고자 했으나, 가장 상단엔 index가 있었다. 그래서 11개를 가져와서 필요한 데이터에 공통으로 들어간 첫글자 “A”가 일치하는지 확인 후에 sed로 변경해주었다. 공백을 기준으로 단어 4개를 받아오면 각각 ‘Aug’, ‘2018’, ‘dd’, ‘-‘ 이다. 2018과 ‘dd’, ‘Aug의 순서를 바꿔주고, Aug는 08로 변환해주었다. ‘-‘는 필요없기 때문에 따로 적어주지 않는다면 사라져 보이지 않게 된다.

1. Case 6

6번 기능은 팀을 고르면 그 팀이 홈에서 가장 큰 득점 차이로 이긴 경기들을 모두 출력하는 것이었다. 팀 순서가 teams.csv에 적힌 순서이므로 index를 제거한 NR-1로 받아와 hometeam에 팀명을 저장해둔다. 그 후에 새로운 파일 match.txt를 만들어서 해당 팀의 모든 경기의 득실차이를 계산하고, 내림차순으로 정렬하여 저장한다. 가장 위에 있는 데이터를 가져오면 가장 큰 득실차이이므로, highnum에 저장하여 matches.csv에서 $hometeam과 같은 문자열이 home\_team\_name인 경기 중에 득실차이를 계산하여 $highnum과 같다면 출력한다. 중요한 것은 이후 match.txt파일을 지워줘야 한다는 것이다. 그렇지 않는다면 match.txt 파일에 계속 데이터가 쌓이게 되어 용량 손실이나 결과값이 바뀔 수 있다.

1. Case 7

Bye!가 출력되게만 하면 until문에 의해 자동으로 종료된다.

1. 총평

이론으로만 공부하는 것보다 이렇게 직접 프로젝트를 진행하면서 프로그램을 만들어보니 훨씬 더 피부에 와닿는 기분이다. 통계자료를 직접 활용하여 이것저것 다루어보고 출력해보는 것이 신선한 경험이었다. 스스로는 통계를 다루거나 기본적인 프로그램을 만들어본 적이 없었기 때문에 2학년 과정을 밟아가는 중에 큰 전환점이 될 수 있을 것 같다. 다만, case 4와 case 5에서의 sort 명령어 옵션 구성이 조금 다른데, 이것은 아직 해결하지 못한 채로 제출하게 되는 것이 조금 아쉽다. 실행은 잘 되는 것 같아 다행인데, case 5에서 case 4처럼 옵션을 주게 된다면 맨시티의 경우에서만 최다 득점 선수가 출력되지 않는 상황이 생긴다. 이런 아쉬움이 남아 제출에 확신은 없지만, 많은 걸 알아가게 된 것 같아 스스로는 만족스럽다. 앞으로 리눅스를 사용할 때 더 재밌게 활용할 수 있을 것 같다.