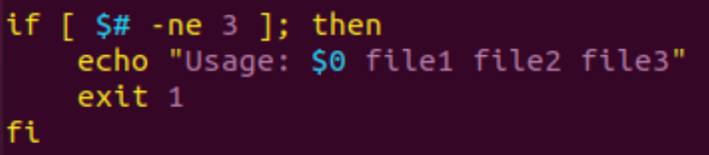
오픈소스SW개론 Project 1

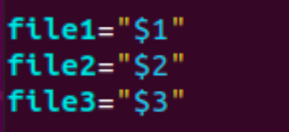
|  |  |  |  |
| --- | --- | --- | --- |
| 분반 | AIE2009-001 | 과 | 인공지능공학과 |
| 학번 | 12224383 | 이름 | 김가은 |



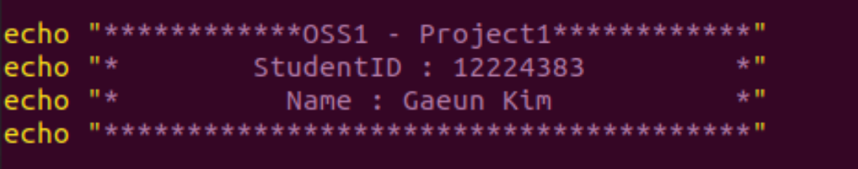
쉘 스크립트를 해석할 때 bash가 사용되도록 지정해주는 SheBang 라인이다.



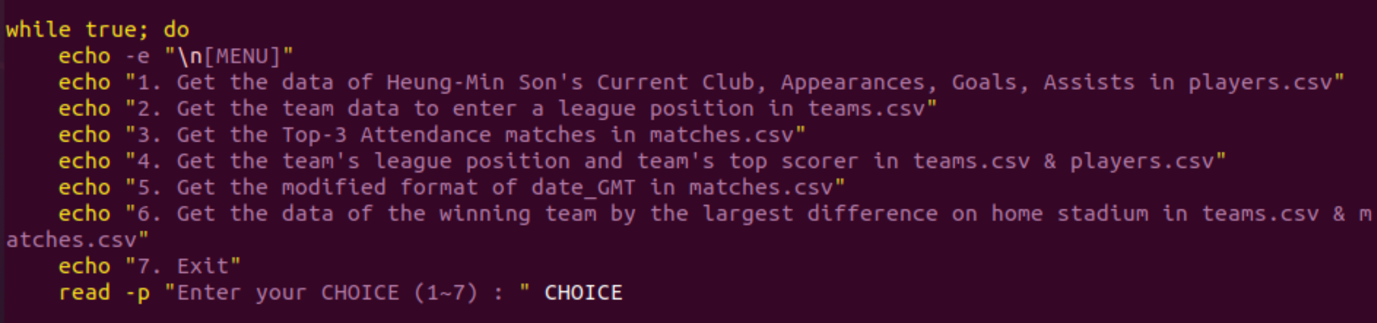
파일 세 개가 argument로 입력되어야 하므로 argument의 개수를 의미하는 ‘$#’ 가 3이 아닐 경우에 스크립트 명을 나타내는 $0을 포함한 에러 메시지를 출력하고 실행을 중단하도록 하였다.



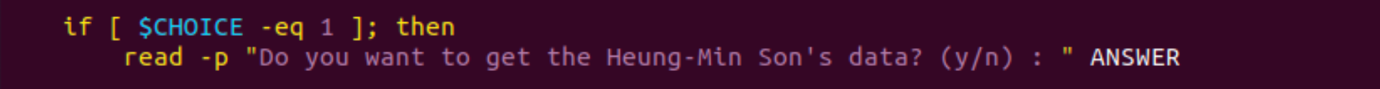
argument를 위치 파라미터인 ‘file1’, ‘file2’, ‘file3’에 지정한다.



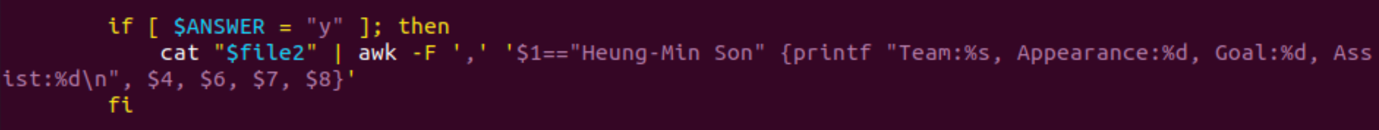
학생 정보를 출력한다.



사용자가 종료 옵션인 7을 입력할 때까지 반복되는 while문을 시작한다. 메뉴를 출력한 후 사용자의 선택을 입력 받고 ‘CHOICE’ 변수에 저장한다.

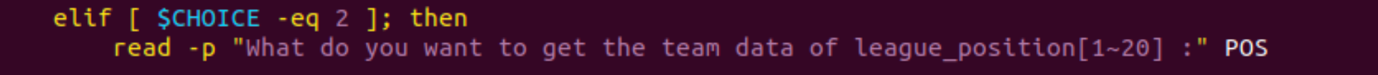


사용자가 1번을 선택했을 경우 손흥민의 데이터를 받을 것인지 묻고 ‘ANSWER’ 변수에 저장한다. ‘ANSWER’가 “y”일 때만 다음 명령을 실행한다.

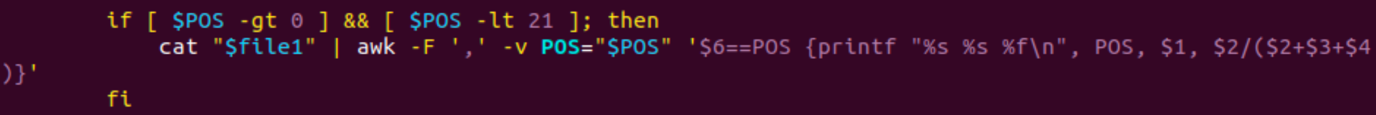


‘file2’인 ‘players.csv’를 읽고, pipe로 그 결과를 이어받아 awk 명령을 실행하는데, -F ‘,’ 옵션은 쉼표로 필드를 구분하겠다는 것을 의미한다. 그 후 첫번째 필드가 “Heung-Min Son”인 라인에 대해 printf명령을 실행한다. ‘%s’, ‘%d’는 뒤의 필드 변수가 들어갈 자리에 출력 형식을 나타낸다. ‘$4’, ‘$6’,…은 그 라인의 4번째, 6번째, 7번째, 8번째 필드인 ‘Team’, ‘Appearance’, ‘Goal’, ‘Assist’를 의미한다.

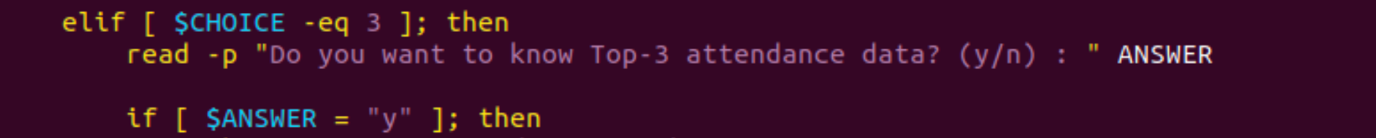
fi는 if문의 끝을 나타내며, if문의 범위를 알려준다.



사용자가 2번을 선택했을 경우 랭킹에 따른 팀 데이터를 볼지 묻고 입력한 랭킹을 POS 변수에 저장한다. POS가 1부터 20 사이일 때 다음 명령을 시행한다.



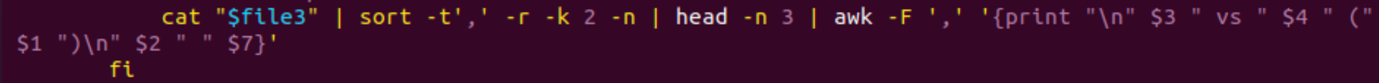
‘file1’인 ‘teams.csv’를 읽고, awk 명령을 실행한다. -v 옵션을 통해 awk에 ‘POS’ 값을 전달한 후, 6번째 필드인 ‘league\_position’이 일치할 경우 ‘POS’ 값과 팀 명, 골 성공률을 출력한다.



사용자가 3번을 선택했을 경우 ‘attendance’가 가장 큰 세 경기를 알려줄지 묻고 ‘ANSWER’ 변수에 저장한다. ‘ANSWER’가 “y”일 때만 다음 명령을 실행한다.

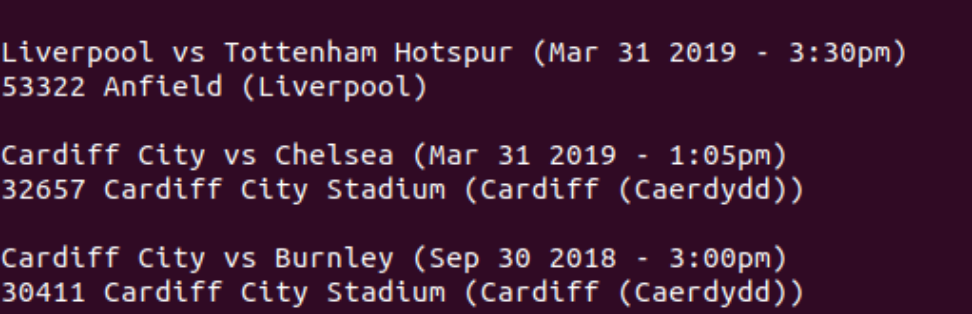


뒤에 출력하는 데이터가 무엇인지 명시하는 라인을 출력한다.



cat 구문을 이용해 ‘file3’인 ‘matches.csv’를 읽는다.

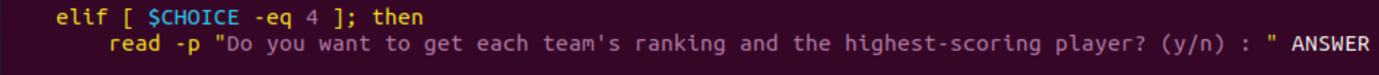
sort할 때는 -t‘,’ 옵션을 이용해 필드를 ‘,’로 구분하는데, 코드를 작성하는 과정에서 ‘,’를 기준으로 구분하지 않으면’matches.csv’ 안의 ‘date\_GMT’, ‘home\_team\_name’, ‘away\_team\_name’, ‘stadium\_name’은 한 필드 안에 띄어쓰기가 포함되어 있는 경우가 있으므로 ‘date\_GMT’ 안의2번째 필드인 ‘일자’를 기준으로 정렬하여 올바른 결과를 출력하지 않는 다는 것을 알게 되었다. 다음이 -t ‘,’를 사용하지 않았을 때의 실행 결과이다. 가장 큰 ‘attendance’가 아니라 가장 큰 ‘일자’인 ’31, 31, 30’ 순서로 출력하는 것을 확인할 수 있다.



또한, 이 옵션은 sort 커맨드에서만 적용되므로 이후의 커맨드에서 다시 ‘,’로 필드를 구분함을 명시해주어야 한다. ‘-r’ 옵션은 reverse, 즉 내림차순으로 정렬함을 의미한다. ‘-k’ 옵션은 정렬을 위한 키를 나타내는데, 2번째 필드인 ‘Attendance’를 기준으로 정렬함을 의미한다. ‘-n’은 숫자로 기준으로 정렬을 수행함을 명시하는 옵션인데, 이 옵션을 사용하지 않을 경우 정렬할 값을 문자열로 인식하여 올바르게 정렬을 하지 못한다. 그리고 이 옵션을 사용하면 각 열의 제목을 나타내는 첫번째 행은 숫자가 아니므로

‘head’ 커맨드는 맨 앞을 기준으로 ‘-n’ 옵션으로 출력할 라인의 개수를 명시한다. Top 3이므로 첫번째 세 줄을 출력하게 한다.

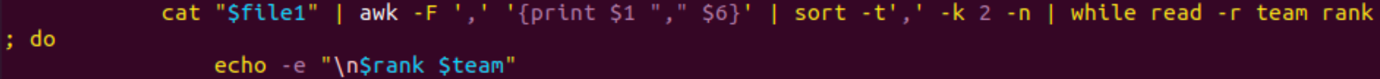
그 후 앞의 명령들을 수행한 결과를 출력하는데, 띄어쓰기, 줄 바꿈을 포함한 문자열은 쌍따옴표 안에 넣고 변수들은 ‘$필드 수’로 따옴표 없이 출력한다.



사용자가 4번을 선택했을 경우 각 팀의 랭킹과 가장 많은 점수를 낸 선수를 볼지 묻고 입력을 ‘ANSWER’에 저장한다. ‘ANSWER’가 “y”일 경우 다음을 실행한다.



‘IFS=,’를 이용해 필드를 ‘,’로 구분하기로 설정하는데, 이전에 각 명령 안에서 옵션을 통해 구분했던 것과 달리 밖에서 구분하는 이유는 ‘while’, ‘read’문과 함께 사용해 각 팀에 대해 ‘teams.csv’와 ‘players.csv’에서 정보를 번갈아 추출하고 출력하기 위해서이다.

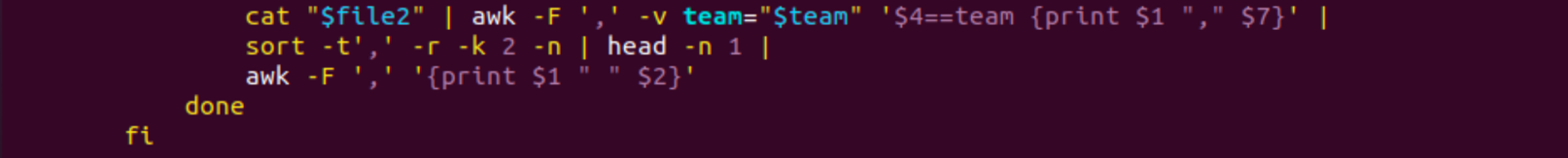


먼저 ‘file1’인 ‘teams.csv’에서 사용할 필드인 $1(팀 명), $6(랭크)을 쉼표로 구분해 출력한다. 이렇게 하는 이유는 read 명령에서 사용하지 않을 필드는 미리 제외하기 위함이다. 미리 제거하지 않는다면 “read -r team \_ \_ \_ \_ rank \_ \_ \_”와 같이 사용하지 않는 필드를 ‘\_’로 지정해줘야 한다. 또, 쉼표로 구분해서 출력해야 pipe로 이어지는 구문에서 쉼표를 기준으로 필드를 구분할 수 있게 된다. 따라서 이렇게 pipe로 연결된 구문 안에서는 매번 필드를 쉼표로 구분하고 마지막으로 출력하는 라인에 도달하기 전까지는 쉼표를 넣어 출력한다.

$1, $6만으로 구성된 라인들을 sort 명령을 이용해 정렬하는데, 넘겨받은 라인에서 $6은 두번째 필드이므로 ‘-k 2’로 설정하여 진행한다.

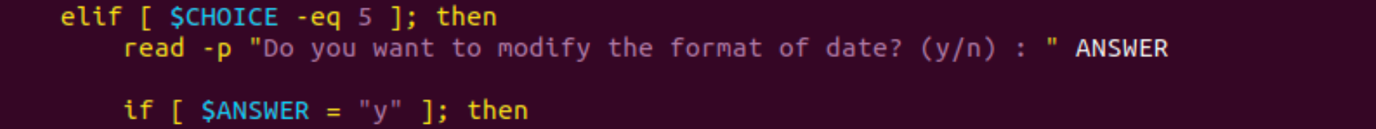
그렇게 넘겨 받은 데이터를 read 커맨드는 한 줄씩 읽고 각 필드를 ‘team’, ‘rank’ 변수에 저장한다. 여기서 ‘-r’ 옵션은 ‘\’ 문자를 이스케이프 문자가 아니라 문자 그대로 해석하게 한다. while문은 파일의 끝에 도달할 때까지 한 줄씩 읽고 그에 대해 while문 안에 있는 다음 구문들을 실행한다.

먼저 ‘teams.csv’에서 구한 팀 명과 순위를 출력한다. 그리고 ‘file2’인 ‘players.csv’에서 가장 많은 골을 기록한 선수를 구해 출력한다.



awk 커맨드를 이용해 해당 팀에 소속된 선수들을 구하기 위해 -v 옵션을 이용해 ‘team’ 변수에 팀 명을 저장하고, 4번째 필드인 팀 명이 ‘team’과 일치할 경우에 그 라인의 $1인 선수명과 $7인 골 수를 쉼표로 구분해 출력한다. 이때, 숫자가 아닌 행의 이름이 들어있는 첫번째 줄은 ‘$4==team’을 충족할 수 없으므로 ‘NR!=1’ 조건을 추가하지 않았다.

그 결과를 이어 받아 2번째 필드인 골 수를 기준으로 내림차 순으로 정렬한 후, 가장 위에 있는 줄에서 선수 명과 골 수를 공백으로 구분해 출력한다. 이후에 필드를 구분할 일이 없는 마지막 명령이므로 쉼표로 구분할 필요 없이 화면에 출력되어야 하는 형식으로 출력하고, while문이 끝났음을 명시하는 ‘done’이 있다.



사용자가 5번을 선택했을 경우 날짜의 형식을 바꾸고 싶은지 묻고 ‘ANSWER’에 답을 저장한 후, 답이 “y”일 경우에 다음 명령을 실행한다.



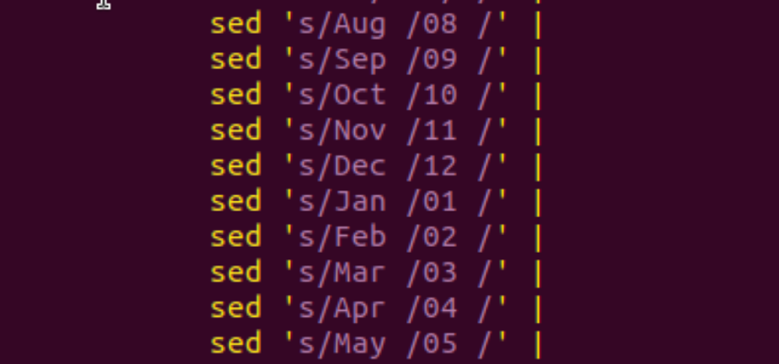
먼저, 데이터가 들어 있는 첫 10개의 줄 만을 출력하기 위해 첫번째 행이 아니고 12번째 행 미만인 조건을 충족할 때 첫번째 필드인 날짜를 출력한다. 그 날짜는 다음과 같은 형식이다.

Aug 10 2018 – 7:00pm



sed 명령 중 특정 부분을 지정한 문자로 대체하는 ‘s/’ 명령을 이용해 “- ” 부분을 “ ”(공백 하나)로 바꾸면 다음과 같다.

Aug 10 2018 7:00pm



sed 구문을 이용해 데이터에 포함되지 않은 달인 6월, 7월을 제외하고, 문자열로 표시된 달을 두자리 형식의 숫자로 바꾸면 다음과 같다.

08 10 2018 7:00pm



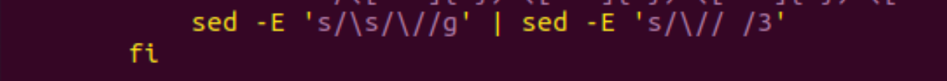
이제 각각의 그룹이 새로운 형식에 맞게 변경되었으므로, 그 순서를 바꾸기 위해 sed 구문을 이용하여 각각의 그룹을 구분하고 맞는 순서로 바꾼다.

‘([0-9] {2})’, ‘([0-9] {4})’에서0부터 9까지의 숫자와 일치하는 문자를 각각 2자리, 4자리로 묶어서 그룹화 한다.

‘([0-9]+:[0-9]{2}[a-z]m)’에서는 ‘시’는 한 자리이거나 두 자리일 수 있으므로 1개 이상임을 의미하는 ‘[0-9]+’로 인식하고, ‘:’은 항상 존재하므로 괄호가 필요 없다. ‘분’은 항상 두 자리이므로 ‘[0-9]{2}’로 인식한다. pm/am에서 ‘m’은 고정, ‘p’와 ‘m’만 바뀌므로 ‘[a-z]’로 알파벳 소문자를 인식한다.

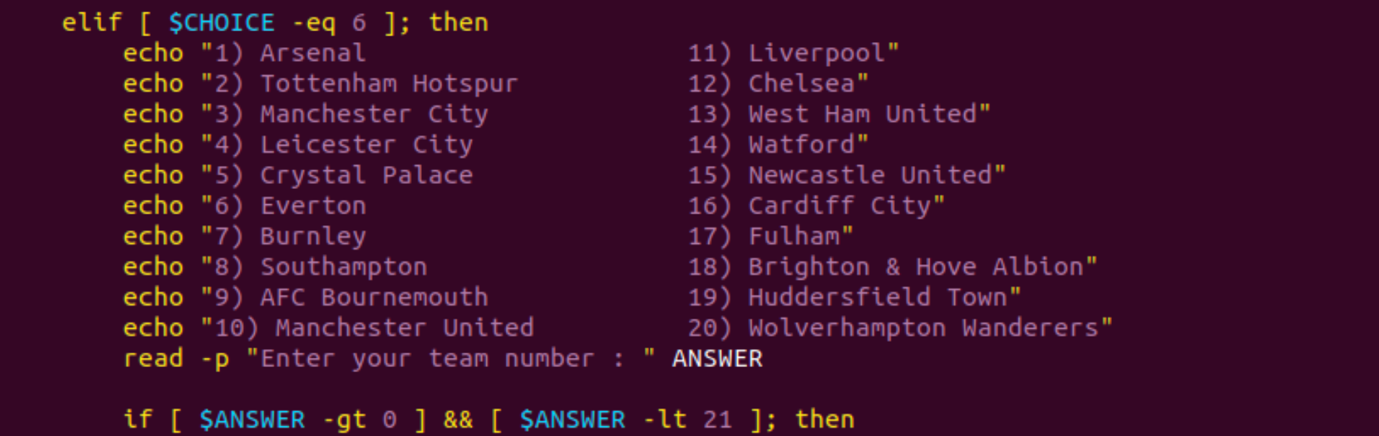
이렇게 각각의 그룹을 지정한 후, ‘\숫자’를 이용해 각 그룹의 순서를 ‘\3 \1 \2 \4’로 바꾼 결과는 다음과 같다.

2018 08 10 7:00pm

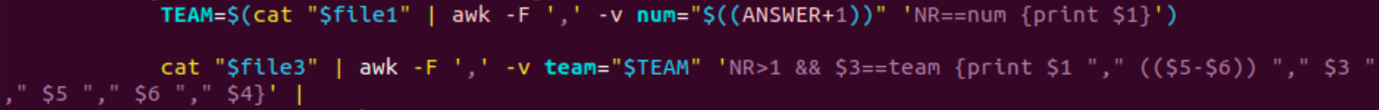


마지막으로 공백을 ‘/’로 대체하기 위해 ‘s/\s/\//g’ 구문을 사용하는데, 여기서 ‘\s’는 공백을 바꾸겠다는 의미이고, ‘\/’는 ‘/’로 바꾸겠다는 의미인데, ‘/’가 문자라는 의미로 ‘\’를 더한 것이고, ‘g’는 모든 공백을 바꾸겠다는 의미이다. 이렇게 먼저 모든 공백을 ‘/’로 바꾼 후 ‘s/\// /3’을 이용해 날짜와 시간 사이의 ‘/’인 3번째 ‘/’만 다시 공백으로 바꾼다. 그에 따른 결과는 다음과 같이 형식이 올바르게 바뀐 모습이다.

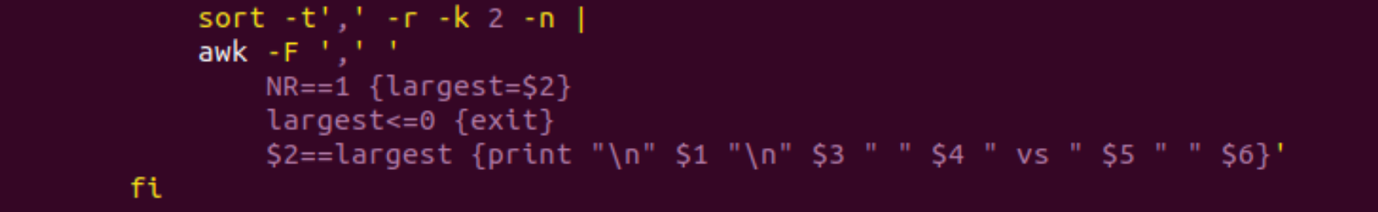
2018/08/10 7:00pm



사용자가 6번을 선택한 경우, 먼저 선택할 수 있는 팀의 번호와 이름을 모두 출력한 후, 팀 번호를 입력 받아 ‘ANSWER’에 저장한다. ‘ANSWER’가 1과 20 사이일 경우에 다음 구문을 실행한다.

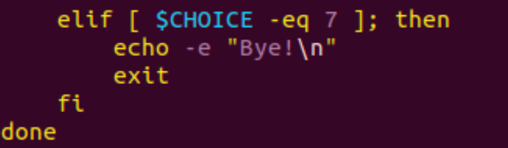


먼저 file1인 ‘teams.csv’에서 번호에 따른 팀 명을 추출해 ‘TEAM’ 변수를 초기화한다. 이를 위해 awk 명령에서 ‘num’ 내장 변수에 팀 명이 있는 행을 할당하는데, ‘teams.csv’의 두 번째 행부터 팀 데이터가 있으므로 ((ANSWER+1)) 식의 결과가 선택한 팀의 이름이 할당되어있는 행의 번호이다. 그 후 행의 번호가 ‘num’과 일치할 경우에 팀 명, ‘(($5-$6))’인 홈팀 기준의 점수차, 홈팀 명, 홈팀 점수, 원정팀 점수, 원정팀 명을 쉼표로 구분하여 출력한다.



그 결과를 두 번째 필드인 점수차를 기준으로 내림차순으로 정렬한다.

정렬한 결과 첫번째 줄이 가장 큰 점수차인데, 같은 최대 점수차의 경기가 더 있을 수 있으므로 awk 커맨드를 이용해 첫번째 줄의 점수차를 ‘largest’ 내장 변수에 저장한다. ‘largest’가 0인 경우, 이겼다고 할 수 없고, 0보다 작은 경우 진 것이므로 ‘largest<=0’일 때 출력하지 않고 명령을 나간다. 점수차가 최대 점수차와 같은 모든 경기를 형식에 맞게 출력하도록 한다.



사용자가 7번을 선택한 경우, 종료 옵션이므로 “Bye!”를 출력하고, 줄바꿈을 한 후, 쉘 스크립트를 종료하도록 한다. 그리고 사용자의 선택 옵션에 대한 ‘if’ 구문이 끝났으므로 ‘fi’로 표시해준다. ‘done’구문은 종료할 때까지 반복되는 while문의 끝을 표시한다.