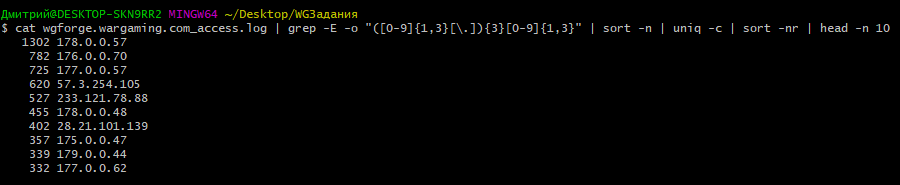
Задание 2.

cat wgforge.wargaming.com\_access.log | grep -E -o "([0-9]{1,3}[\.]){3}[0-9]{1,3}" | sort -n | uniq -c | sort -nr | head -n 10



С помощью grep-функции по шаблону регулярного выражения выбираем айпишники, после чего сортируем их, группируем одинаковые с подсчётом количества одинаковых, ну а далее берём самые многочисленные.

Задание 3.

Запрос:

select Departments.DEPARTMENT\_NAME,

EmpOutSide.NAME

from Departments

inner join Employees as EmpOutSide on Departments.DEPARTMENT\_ID = EmpOutSide.DEPARTMENT\_ID

where SALARY > (select Max(salary) from Employees where lead = 1 and Employees.DEPARTMENT\_ID = EmpOutSide.DEPARTMENT\_ID)

and

lead = 0

Данный запрос выводит сотрудников тех отделов, в пределах которых у них (сотрудников) зарплата больше, чем у их боссов в пределах того же отдела. Если боссов в отделе несколько, то для сравнения с зарплатами простых работяг берётся зарплата самого высокооплачиваемого. Если боссов нет, то ни один сотрудник из этого отдела выведен не будет.

Задание 4.

После 5-часового мозгового марафона я пришёл к выводу, что, по сути, данный скрипт представляет собой завуалированный вариант команды pwn3d!. Данная команда никакого смысла не несёт и среда bash вполне логично выдала бы ошибку, но, к сожалению, кто-то заблокировал поток вывода ошибок командой 2>dev/null. Убрав эту команду, я и узнал, что весь этот скрипт пытается выполнить несуществующую команду. Команда эта получается путём интерпретации чисел 112 119 110 51 100 33, как кодов символов таблицы ASCII. Этот эффект достигается применением следующей функции:

\_\_() { awk "BEGIN{printf \"%c\", $1}"; };

Эта функция использует встроенный метод printf с модификатором %с, ожидающим на вход символ. Получая на вход число, она воспринимает его, как код символа и выводит именно этот символ.

Далее следующая строка:

echo '112 119 110 51 100 33'| \

( \_\_\_=0;\_\_\_\_=${\_:-s};\_\_=$\_\_\_\_;\_\_\_\_=${\_\_\_\_}e;\_\_=${\_\_}/;

\_\_\_\_=${\_\_\_\_}d;\_\_\_=$(($\_\_\_+1));

\_\_="${\_\_}\\([0-9]\\+\\)\\s\\?/\$(\_\_ \\${\_\_\_})/g";$\_\_\_\_$IFS"$\_\_")

сама по себе возвращает комбинацию команд $(\_\_ 112)$(\_\_ 119)$(\_\_ 110)$(\_\_ 51)$(\_\_ 100)$(\_\_ 33). После чего команда eval считывает результат (нашу заветную строку). В результате работы функций и выходит слово pwn3d!. Это слово, как уже говорилось ранее, не имеет смысла. Помозговав ещё 3 часа я полностью разобрал принцип работы скрипта. Во входном тексте он заменяет числа по следующему правилу: число => (\_\_ число). При этом пробелы справа от чисел игнорируются и как бы съедаются при замене. Но этого не происходит с другими символами. Поэтому мне пришла в голову идея подставить в строку списка аргументов echo букву d между числами 119 и 110. Вот так выглядел итоговый скрипт:

#!/bin/bash

# To solve this task you must put ONE word somewhere to get this script working!

\_\_() { awk "BEGIN{printf \"%c\", $1}"; };

eval $(echo '112 119 d 110 51 100 33'| \

( \_\_\_=0;\_\_\_\_=${\_:-s};\_\_=$\_\_\_\_;\_\_\_\_=${\_\_\_\_}e;\_\_=${\_\_}/;

\_\_\_\_=${\_\_\_\_}d;\_\_\_=$(($\_\_\_+1));

\_\_="${\_\_}\\([0-9]\\+\\)\\s\\?/\$(\_\_ \\${\_\_\_})/g";$\_\_\_\_$IFS"$\_\_")) 2>/dev/null;

Результатом работы команды eval была команда pwd n3d!. Команда pwd выводила расположение текущей директории, а аргумент n3d! игнорировался.

Задание 5.