ΠΛΗ211
ΕΡΓΑΛΕΙΑ ΑΝΑΠΤ. ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΩΝ
ΑΝΑΦΟΡΑ ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ 1
Σπουδαστές:

Καρτάκης Άγγελος:2015030042 (Α)
Παντελής Κωνσταντίνος:2015030070 (Β)

Η συγγραφή κώδικα πραγματοποιήθηκε ταυτόχρον, ωστόσο ο (Α) ασχολήθηκε κυρίως με την αλγοριθμική επίλυση και ο (Β) με το πλαίσιο των αρχείων και τη λήψη πληροφοριών απο αύτά.

Σκοπός Άσκησης:

Η εξοικείωση με το κέλυφος προγραμματισμού Bash (Bourne Again Shell) με το λειτουργικό σύστημα Linux καθώς και η πλοήγηση και επεξεργασία αρχείων μέσω απλών εντολών κονσόλας σε αυτό (Linux).

Η εργασία οργανώθηκε σε δύο μέρη (προγράμματα – scripts).

- ***Regr***.
    - Πολλαπλά αρχεία εισόδου.
    - Υπολογισμός παραμέτρων γραμμικής παλινδρόμησης για το κάθε ένα.
    - Εκτύπωση στην έξοδο μορφοποιημένου μηνύματος για κάθε αρχείο.

- ***Results.***
    - Ένα αρχείο εισόδου, αποτελέσματα αγώνων μεταξύ ομάδων.
    - Υπολογισμός βαθμολογίας και στατιστικών.
    - Εκτύπωση στην έξοδο.

Περιγραφή Ζητουμένων – Προσδιορισμός Συνθηκών:

Μέρος Α: “Regr”

Δοθέντων Ν αρχείων όπου η πληροφορία οργανώνεται σε γραμμές της μορφής <numX>:<numY> ζητούνται να υπολογισθούν οι παραμέτροι γραμμικής παλινδρόμησης καθώς και το σφάλμα για την κάθε συλλογή.

Η υλοποίηση της λύσης θα πρέπει να υποστηρίζει αρχεία με άγνωστο – ακανόνιστο αριθμό γραμμών, ενώ τυχών διαφορές στη μορφή της καθορισμένης εισόδου προκαλούν σφάλμα εκτέλεσης (π.χ αριθμοί να χωρίζονται απο “-” αντί για “:”). Τέλος τα αρχεία δίνονται απο τον εκτελεστή του script μέσω της κονσόλας, ενώ θα πρέπει να υπάρχει κατάλληλος χειρισμός σε περίπτωση μη ύπαρξης τους ή αδυναμίας διαβάσματος αυτών (read_bit = 0).

Αναφορικά με τους συντελεστές α,β ζητήθηκε τα δεκαδικά τους ψηφία (εφόσον υπάρχουν) να περιορισθούν σε 2.

Η έξοδος στην κονσόλα πρέπει να ακολουθεί την μορφή:

FILE:<fileName> a=<someDecimal>,b=<someDecimal>,err=<someDecimal>

Μέρος Β: “Results”

Δοθέντος ενός αρχείου το οποίο περιέχει αποτελέσματα αγώνων της μορφής ***<teamA>-<teamB>:<scoreA>-<scoreB>*** αγνώστου πλήθους το script θα έπρεπε να διαβάζει τις ομάδες, εκτυπώνει την πλήρη λίστα τους ταξινομημένη ανάλογα με τα αποτελέσματα (με ειδική ρήτρα σε περίπτωση ισοβαθμίας) καθώς και το ρεκόρ γκόλ υπέρ-γκόλ κατά.

Συνοψίζωντας πρόκειται για έξοδο της παραπάνω μορφής:

<position>. <teamName> <pointsGained> <goalScored> <goalConceeded>

(κάθε κενό εκτυπώνεται ως tab)

Δυνητικά σημεία ενδιαφέροντος θα μπορούσαν εδώ να είναι ο χειρισμός των κενών στο αρχείο-είσοδο, η απαλοιφή διπλοτύπων ομάδων (αν υπάρχουν).

Υλοποίηση:

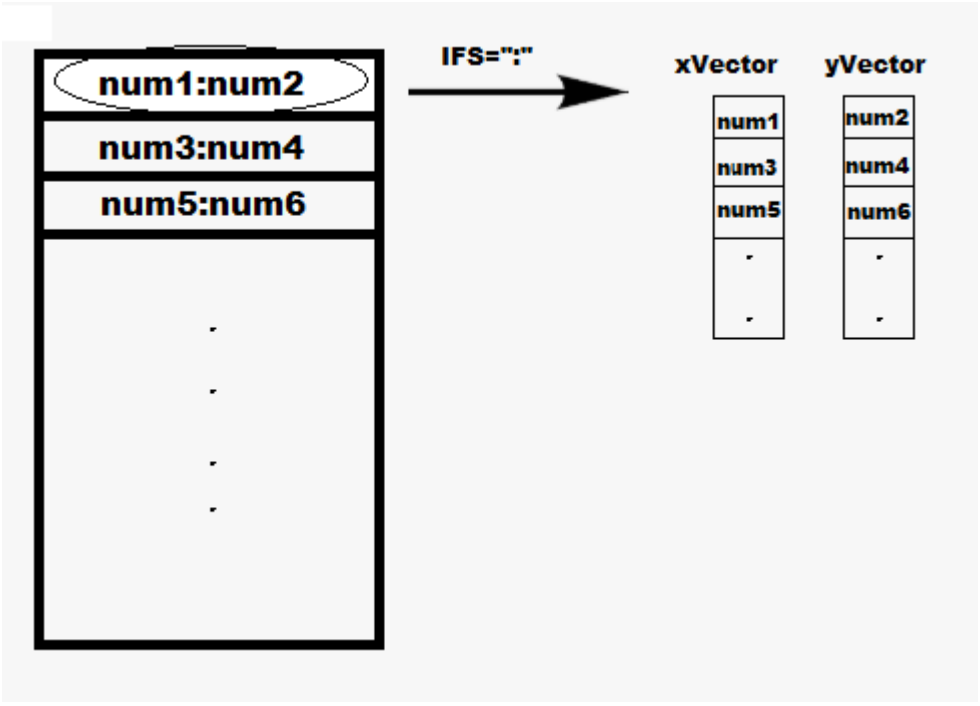
Μέρος Α: “Regr”

Αρχικά λαμβάνονται τα αρχεία ώς ορίσματα απο το command line (κονσόλα), εφόσον κάθε ένα απο αυτά υπάρχει (βρίσκεται στο ίδιο directory με το script ή έχει δωθεί η πλήρης διαδρομή σε αυτό) ελέγχεται το κατά πόσο αυτό το στοιχείο μπορεί να διαβασθεί,αν το read_bit είναι απενεργοποιημένο χρησιμοποιείται η εντολή chmod και μία μάσκα με το 444_{10} το ισοδύναμο του 100100100_2(*). Σε περίπτωση που το αρχείο δεν υπάρχει η λειτουργία του προγράμματος διακόπτεται και μεταβαίνει σε επόμενο αρχείο.

Έπειτα για κάθε αρχείο καθορίζεται το πλήθος γραμμών του με τη βοήθεια της wc (word count, υπάρχει man page) εντολής κι προσπελαύνεται το αρχείο γραμμή προς γραμμή με χρήση και αλλαγμένου IFS ώστε να έχουμε διαθέσιμα τα στοιχεία κάθε διανύσματος (χωρίζονται με “:”).

Αποθηκεύονται στη συνέχεια σε δύο πίνακες τους xVector,yVector, οι οποίοι και προσπελαύνονται καταλλήλως για τον υπολογισμό των υπόλοιπων μεγεθών του δωθέντος τυπολογίου και έπειτα των τελικών συντελεστών.

Σχηματικά η διαδικασία:



Έστω αρχείο 3 γραμμών.

Αξιοπρόσεκτα σημεία:

- ***wc -l $file***: θα επιστρέψει (στο stdout) τις γραμμές και το όνομα του αρχείου.
- ***($(wc -l $file))***: το ίδιο χωρίς εκτύπωση, υπο τη μορφή πίνακα (με το πρώτο στοιχείο να είναι οι γραμμές του αρχείου).
- **<*command1*> | <*command2*>| ..| <*commandN*>** : pipping εντολών, το αποτέλεσμα της πρώτης χρησιμοποιείται στην δεύτερη ή οι εντολές δεξιά προσδιορίζουν την εκτέλεση της αριστερής (**). ***Χρησιμοποίηση πολλαπλών εντολών και μεταφορά δεδομένων μεταξύ αυτών.***
- ***varCom=$command :*** Αποθήκευση του αποτελέσματος εντολής ώς μεταβλητή (όμοιο με type Casting σε C).
- ***(echo " scale=2; (($var1 * $var2) + $var3) / 1 " | bc -l*** : θα εκτελεσθούν οι πράξεις μεταξύ δεκαδικών, το αποτέλεσμα θα αποθηκευθεί - εμφανισθεί με ακρίβεια δύο δεκαδικών ψηφίων μόνο όμως αν γίνει διαίρεση (εξ' ου και διαίρεση με το 1).

Κώδικας αξιοπρόσεκτων σημείων:

1) Έλεγχος ορθής λειτουργίας:

```
files=($@)
for filePath in "${files[@]}"
do

        #check file existance,not emptyness (zero - sized file)

        if [ -e "$filePath" ] && [ -s "$filePath" ]
        then


                #check whether read permission is on for the person executing the script,if not manually open the read bit (msb)

                if [ !  -r "$filePath" ]
                then

                        chmod 444 $filePath     #enable read permission for everyone by a mask with 1's in respective read_bits

                fi


                        #Count the lines of the file given,cat command can also be used here

                        result=($(wc -l $filePath))  #variable type casting will work,then interpreting result as an array
                        lineNum=${result[0]}


        else

                echo "Unable to locate the File inside working dir"
                exit 126

        fi
```

-Έλεγχος ύπαρξης και μηδενικού μεγέθους (άδειου αρχείου)

-Έλεγχος read permission για το αρχείο, αν δεν υπάρχει ανοίγει για όλους.

2) Προσπέλαση και διάβασμα του αρχείου:

```
while read -r tmpLines

do

        lines[i]=$tmpLines
        let "i++"

done < "$filePath"
```

Διάβασμα όλων των γραμμών

```
oIFS=$IFS                                          #keep value of IFS somewhere
IFS=':'
        for (( i = 0; i < "$lineNum"; i++ ))
        do

                ind=0;
                read -ra ff <<< "${lines[i]}"
                for j in "${fff[@]}";
                do

                        if (( $ind % 2 ))          → Στοιχεία με περιττό index ανοίκουν στο
                        then                         xVar, με άρτιο στο yVar. Τα παραπάνω
                                                     εύκολα εξηγούνται και απο το σχήμα.

                                yVar[i]="$j"

                        else


                                xVar[i]="$j"

                        fi
                        let "ind++"


                done


        done
IFS=$oIFS       #restore IFS                       → Επαναφορά default Internal Field
                                                     Separator (IFS), συνήθως " \n\t".
```

3) Μαθηματικές Πράξεις:

Όπως προαναφέρθηκε χρησιμοποιήθηκε η εντολή bc -l (floating point operations with standard math library enabled).Ακολουθούν οι εντολές για τον υπολογισμό των α,β.

```
a=$(echo " scale=2; ( ( $vectorSize * $xySum ) - ( $xSum * $ySum ) ) / ( ( $vectorSize * $xSqSum ) -( $xSum * $xSum ) ) "  | bc -l )

b=$(echo " scale=2; ( $ySum - $a * $xSum ) / $vectorSize " | bc -l)
```

Αξιοόλογη η απαραίτητη ύπαρξη διαίρεσης για να ¨δουλέψει¨ η scale.

Μέρος Β: “Results”

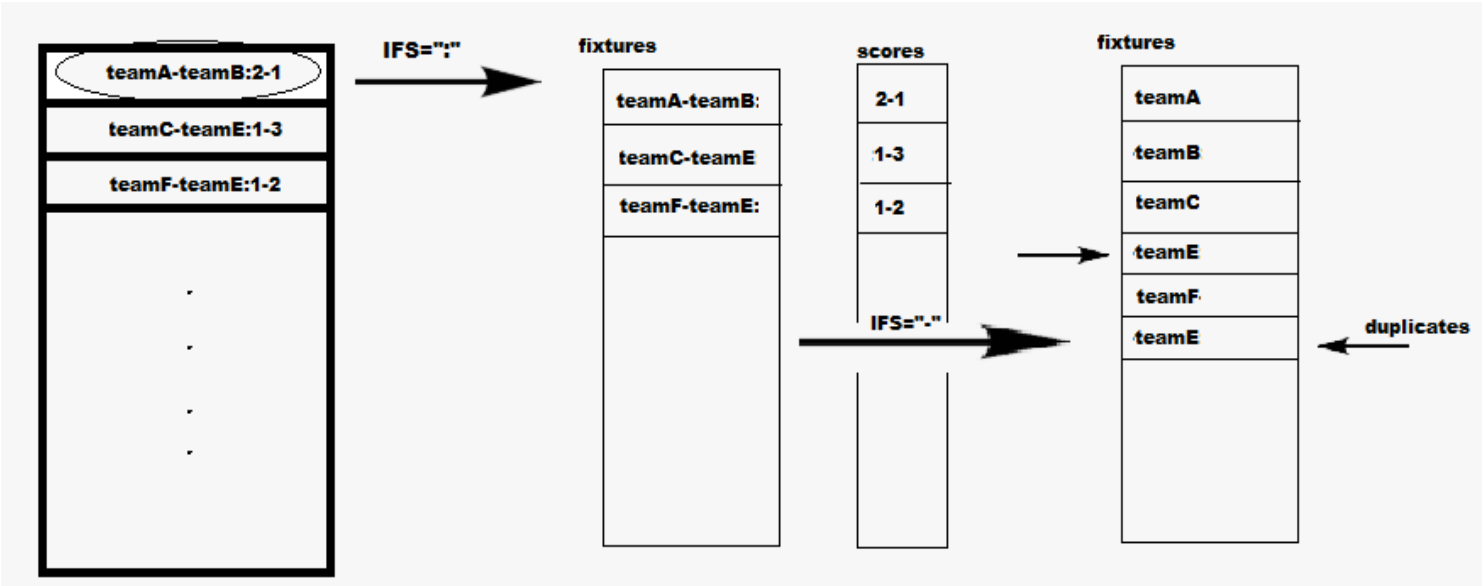
Δημιουργείται αρχείο temp (με την εντολή touch) στο οποίο θα αποθηκευθεί το αρχείο εισόδου με τα όποια κενά να έχουν αντικατασταθεί απο “_”. Το παραπάνω γίνεται καθότι κενά στα ονόματα των ομάδων προκαλούν πρόβλημα στην τελική σύγκριση. Προφανώς οι έλεγχοι ύπαρξης και προσπελασιμότητας αρχείου εδώ είναι ίδιοι με το Μέρος Α.

Έπειτα διαβάζεται η κάθε γραμμή του αρχείου. Γενικά σε αρχείο Ν γραμμών θεωρείται καλή πρακτική να τις διαβάσουμε όλες κάτι που θα κοστίσει Ν προσβάσεις και έπειτα να επεξεργαστούμε την πληροφορία που διαβάσαμε, παρά να διαβάσουμε το αρχείο όσες φορές τύχει (π.χ 2 ή 3 είναι μια λύση εδώ). Απο κάθε γραμμή τροποποιώντας τον IFS μπορούμε να λαμβάνουμε μια λίστα των ομάδων στο αρχείο (τα διπλότυπα είναι εντός εδώ) καθώς και μία λίστα αγώνων και σκόρ. Ταξινομείται ο πίνακας ομάδων ώστε να αφαιρεθούν διπλότυπα.

Έπειτα γεμίζεται ένας associative array, ο οποίος αντιπροσωπεύει την βαθμολογία κάθε ομάδας με τους βαθμούς εκάστης ανάλογα τα σκόρ. Τέλος ταξινομείται (με αλγόριθμο φυσαλίδας) ο πίνακας των ομάδων με κριτήριο την βαθμολογία τους. Πλέον η πρώτη ομάδα στον πίνακα ομάδων είναι και αυτή με τους περισσότερους βαθμούς. Τέλος κατα τη διάρκεια βαθμολόγησης των ομάδων αρχικοποιούντε και οι πίνακες goalScored,goalsConceeded.

Στο τελικό στάδιο εκτυπώνονται οι ομάδες όπου με pipping αντικαθίσταται το “_” με το αρχικό κενό. Επίσης διαγράφεται το προσωρινό αρχείο που χρησιμοποιήθηκε.

Σχημαρικά η διαδικασία:



Αξιοπρόσεκτα σημεία:

- ***($(echo "${teams[@]}" | tr ' ' '\n' | sort -u | tr '\n' ' ')):*** Αντικαθιστά “ ” με “\n” , ταξινομεί και επαναφέρει. Όλα τα παραπάνω αποθηκεύονται ώς πίνακας σε άλλη μεταβλητή. Ένας τρόπος να αφαιρεθούν τα duplicates.
- ***printf '%d.\t%-10s\t%d\t%d-%d\n' :*** χρήση της printf με ειδικούς χαρακτήρες εκτύπωσης.

Κώδικας αξιοπρόσεκτων σημείων:

1) Διάβασμα αρχείου

```
while IFS=:  read -ra tmpLines                    ⟶   Με αυτή τη γραφή δε χρειάζεται “σώσιμο” στον IFS
do



                    #retrieve fixtures and score of each one respectivelly

                    fixtures[i]="${tmpLines[0]}"
                    scores[i]="${tmpLines[1]}"

                    #retrieve teams
                    oIFS=$IFS
                    IFS='-'
                    read -ra ll <<< "${fixtures[i]}"
                    teams[j]="${ll[0]}"
                    let "j++"
                    teams[j]="${ll[1]}"
                    let "j++"


                    let "i++"

                    #restore IFS from inner loop change,outter is auto-retrieved in this syntax

                    IFS=$oIFS



done < "$filePath"
```

2) Διαγραφή διπλοτύπων

```
teamsFinal=($(echo "${teams[@]}" | tr ' ' '\n' | sort -u  | tr '\n' ' '))   ⟶   Το αποτέλεσμα σε νέο πίνακα.
```

3) **Βαθμολόγηση και χρήση Associative Arrays**

```
IFS='-'
for (( i = 0; i < "$fixtureNum"; i++ ))
do

        read -ra kk <<< "${fixtures[i]}"
        teamA=${kk[0]}
        teamB=${kk[1]}


        read -ra ll <<< "${scores[i]}"
        scoreA=${ll[0]}
        scoreB=${ll[1]}

        #interprete variables in a way that helps determine points for each team

        let "goalScored[$teamA]=${goalScored[$teamA]} + $scoreA "

        let "goalConceded[$teamA]= ${goalConceded[$teamA]} + $scoreB "



        let " goalScored[$teamB] = ${goalScored[$teamB]} + $scoreB "
        let " goalConceded[$teamB] = ${goalConceded[$teamB]} + $scoreA"

        #create scoring table ( win -> 3 points, draw -> 1 point )

                if [[ "$scoreA" -gt "$scoreB" ]]
                then


                        let "points[$teamA] = ${points[$teamA]} + 3 "     #teamA won
                        let "points[$teamB] = ${points[$teamB]} + 0 "

                elif [[ "$scoreA" -eq  "$scoreB" ]]
                then

                        let "points[$teamA] = ${points[$teamA]} + 1 "   #draw,each team earns 1 point
                        let "points[$teamB] = ${points[$teamB]} + 1 "

                else


                        let "points[$teamB] = ${points[$teamB]} + 3 "   #teamB won
                        let "points[$teamA] = ${points[$teamA]} + 0 "

                fi



        done
```

Associative Array Usage

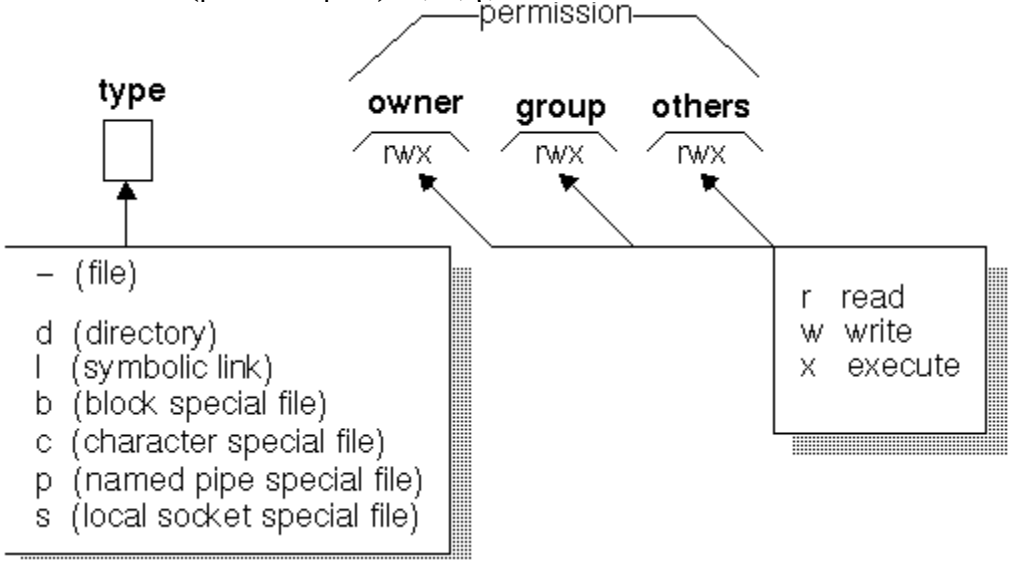
Παράρτημα:

(****):** Σύστημα αρχείων Linux, δικαιώματα πρόσβασης και η εντολή chmod*

Το σύστημα αρχείων στα Linux δίνει τη δινατότητα ελέγχου και οργάνωσης πληροφορίας αναφορικά με το ποιός και σε τι βαθμό μπορει να χρησιμοποιήσει ενα αρχείο μέσω του συστήματος οργάνωσης αδειών.

Στα Linux υπάρχουν 3 κύριοι τύποι χρηστών για κάθε αρχείο (owner,group,others) και 3 κύριοι τύποι αδειών (read,write,execute-x)

Συγκεκριμένα για ένα τυπικό αρχείο έστω ***someFile.txt*** οι πληροφορίες αυτές μπορούν να απεικονισθούν (μία ls -l αρκεί) ώς εξής:



Τα παραπάνω μπορούν να αντιμετωπισθούν ώς ένας καταχωρητής του οποίου ενδιαφέρον έχουν τα 9 lsb, ενώ το 10ο απεικονίζει τον τύπο αρχείου. Κατά την δημιουργία ενός νέου αρχείου (έστω touch afile.txt) οι άδειες αυτές λαμβάνουν μια default τιμή η οποία μπορεί να αλλαχθεί με την εντολή unmask. Κάθε φορά που επιθυμούμε να αλλάξουμε τα δικαιώματα πρόσβασης μια ομάδας χρηστών για ένα αρχείο θα πρέπει να χρησιμοποιήσουμε λοιπόν την chmod.

Η παραπάνω εντολή μπορεί να χρησιμοποιηθεί τόσο συμβολικά όσο και απευθείας με μάσκα στα bits που μας ενδιαφέρουν.

Παράδειγμα:

chmod 754 afile.txt

7 → 111_2 = rwx (owner)
5 → 101_2 = rwx (group of users) **ACTIVE HIGH BITS (1=enabled, 0=disabled)**
4 → 100_2 = rwx (others)

Οπότε ο ιδιοκτήτης του αρχείου (αυτός που το δημιούργησε) μπορεί πλέον τόσο να το διαβάσει(r), όσο και να το επεξεργαστεί(w) αλλά και να το εκτελέσει αν αυτό είναι εκτελέσιμο(x). Οι υπόλοιποι χρήστες μπορούν να εκτελούν το αρχείο καθώς και να το διαβάσουν αλλά όχι να το

τροποποιήσουν αφού το αντίστοιχο bit είναι 0 στη μάσκα, ενώ για όλους του άλλους (π.χ δίκτυο) το αρχείο έχει μόνο δικαιώματα διαβάσματος.