ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
2η Εργασία - Τμήμα Περιττών Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '16
Ημερομηνία Ανακοίνωσης: 27/10
Ημερομηνία Υποβολής: 20/11 και Ώρα 23:59

Εισαγωγή στην Εργασία:

Ο στόχος αυτής της εργασίας είναι να εξοικειωθείτε με την δημιουργία *ιεραρχιών από διεργασίες* χρησιμοποιώντας την κλήση συστήματος fork() που σε συνδυασμό με την κλήση exec*() μπορεί να βοηθήσει να επιτύχουμε διαφοροποίηση και επιμερισμό εργασιών. Επιπροσθέτως, θα χρησιμοποιήσετε διάφορες κλήσεις συστήματος όπως wait, mkfifo, fread, fwrite, για I/Os, απλό συγχρονισμό, και επικοινωνία μεταξύ των διεργασιών.

Η ιεραρχία διεργασιών που θα δημιουργήσετε θα επιλύει συνεργατικά πολλαπλές επερωτήσεις χώρου στις 2-διαστάσεις για ένα σύνολο σημείων που παρέχονται από ένα δυαδικό αρχείο. Οι διαδικασίες του χαμηλότερου επιπέδου στην ιεραρχία αναλαμβάνουν να εκτελέσουν απλά αυτόνομα προγράμματα χρήσης (utilities) που επιτελούν συγκεκριμένες επερωτήσεις χώρου.

Πιο κάτω πρώτα περιγράφουμε αυτά τα utilties και κατόπιν σκιαγραφούμε την συνολική λειτουργικότητα του προγράμματος shapes που θα αναπτύξετε και που δημιουργεί την αναμενόμενη ιεραρχία διεργασιών. Τέλος παραθέτουμε τα αναμενόμενά αποτελέσματα εξόδου των διεργασιών και του προγράμματος shapes.

Προγράμματα Χρήσης (Utilities):

Θα πρέπει να αναπτύξετε πέντε απλά προγράμματα χρήσης (utilities), τα οποία υλοποιούν χωρικές επερωτήσεις στον δισδιάστατο χώρο (2-dimensional spatial queries). Όλα τα utilities μπορούν να εκτελεσθούν με τα εξής ορίσματα:
./utilityX -i InputBinaryFile -o OutputFile -a UtilityArgs [-f Offset] [-n PointsToReadCount]
όπου:

- utilityX είναι το εκάστοτε από τα πέντε utilities που θα υλοποιήσετε,
- InputBinaryFile είναι το δυαδικό αρχείο με τα δεδομένα εισόδου γραμμικοποιημένα (serialized) σε μορφή float, float, ... (χωρίς κενό ή κόμμα μεταξύ τους),
- OutputFile το αρχείο εξόδου, που θα περιέχει σε μορφή ASCII τα σημεία τα οποία ικανοποιούν το ερώτημα του εκάστοτε utility, με το κάθε σημείο ανά γραμμή και τις συντεταγμένες του διαχωρισμένες με *TAB*.
- UtilityArgs τα ορίσματα του εκάστοτε utility (τα οποία αναφέρονται αναλυτικά παρακάτω) διαχωρισμένα με κενό μεταξύ τους,
- Offset είναι ο αριθμός των bytes που θα αγνοηθούν από την αρχή του αρχείου. Η εν λόγω παράμετρος είναι προαιρετική και για το λόγο αυτό περιβάλλεται από τις παρενθέσεις [ ],
- PointsToReadCount είναι το πλήθος των σημείων (ζεύγη από float αριθμούς) τα οποία θα διαβάσει το utility από το αρχείο εισόδου. Η εν λόγω παράμετρος είναι προαιρετική και για το λόγο αυτό περιβάλλεται από τις παρενθέσεις [ ].

Οι σημαίες -i/ -o/ -a/ -f/ -n μπορούν να χρησιμοποιηθούν με οποιαδήποτε σειρά στην γραμμή εκτέλεσης του προγράμματος. Το κάθε ένα από τα utilities επιδέχεται περαιτέρω εξειδίκευση μέσω της παραμέτρου -a ανάλογα με την λειτουργία του, ως εξής:

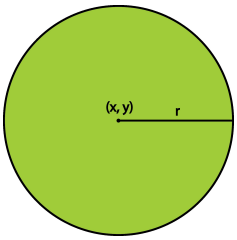
1. circle -a x y r [1]
   Δεδομένου ότι το πρόγραμμα χρήσης είναι το circle, ελέγχεται αν το εκάστοτε σημείο εισόδου ανήκει στον κύκλο ([x,y], r).

[1] https://en.wikipedia.org/wiki/Circle

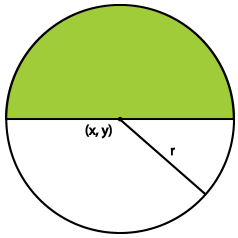
2. semicircle -a x y r N/S/W/E [2]
   Δεδομένου ότι το πρόγραμμα χρήσης είναι το semicircle, ελέγχεται αν το εκάστοτε σημείο εισόδου ανήκει στον κύκλο ([x,y], r) και συγκεκριμένα στο North, South, West, East ημικύκλιο.
3. ring -a x y r1 r2 [3]
   Αν το utility είναι το ring τότε κάθε σημείο εισόδου ελέγχεται αν ανήκει στον δακτύλιο ([x,y], r1, r2).
4. square -a x y r [4]
   Αν το utility είναι το square τότε κάθε σημείο εισόδου ελέγχεται αν ανήκει στο τετράγωνο ([x,y], r) το οποίο έχει κατακόρυφη και οριζόντια διαγώνιο.
5. ellipse -a h k α β [5]
   Δεδομένου ότι το πρόγραμμα χρήσης είναι το ellipse τότε κάθε σημείο εισόδου ελέγχεται αν ανήκει στην έλλειψη που ορίζεται από το κέντρο C(h,k), έχει μεγάλο άξονα τον x και μικρό τον y και της οποίας η εξίσωση είναι η εξής: $\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$.

Οι σειρά που οι παράμετροι παρέχονται στα παραπάνω utilities είναι αυστηρή (δηλ. αλλαγή σειράς υποδηλώνει και ένα διαφορετικό δισδιάστατο χώρο).

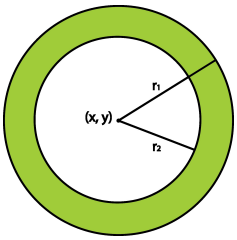
Παρακάτω, βλέπετε τις επερωτήσεις οπτικοποιημένες σε πλήρη αντιστοιχία με την λίστα παραπάνω:



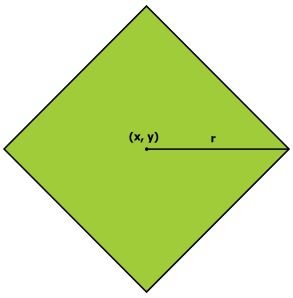
Σχήμα 1: circle x y r



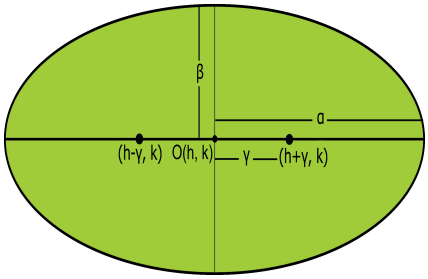
Σχήμα 2: semicircle x y r N



Σχήμα 3: ring x y r1 r2



Σχήμα 4: square x y r



Σχήμα 5: ellipse h k α β

Για παράδειγμα, μια κλήση του utility semicircle είναι η εξής:
prompt> ./semicircle -i InputBinaryFile -o OutputFile -a 2.0 4.0 5.5 W
όπου ψάχνουμε όλα τα σημεία του βρίσκονται στο αρχείο InputBinaryFile αν ανήκουν στο Δυτικό ημικύκλιο που έχει κέντρο το σημειου 2.0 4.0 και ακτίνα 5.5 και τα εκτυπώνουμε στο αρχείο εξόδου OutputFile..

[2] https://en.wikipedia.org/wiki/Semicircle
[3] https://en.wikipedia.org/wiki/Annulus_(mathematics)
[4] https://en.wikipedia.org/wiki/Square
[5] https://en.wikipedia.org/wiki/Ellipse

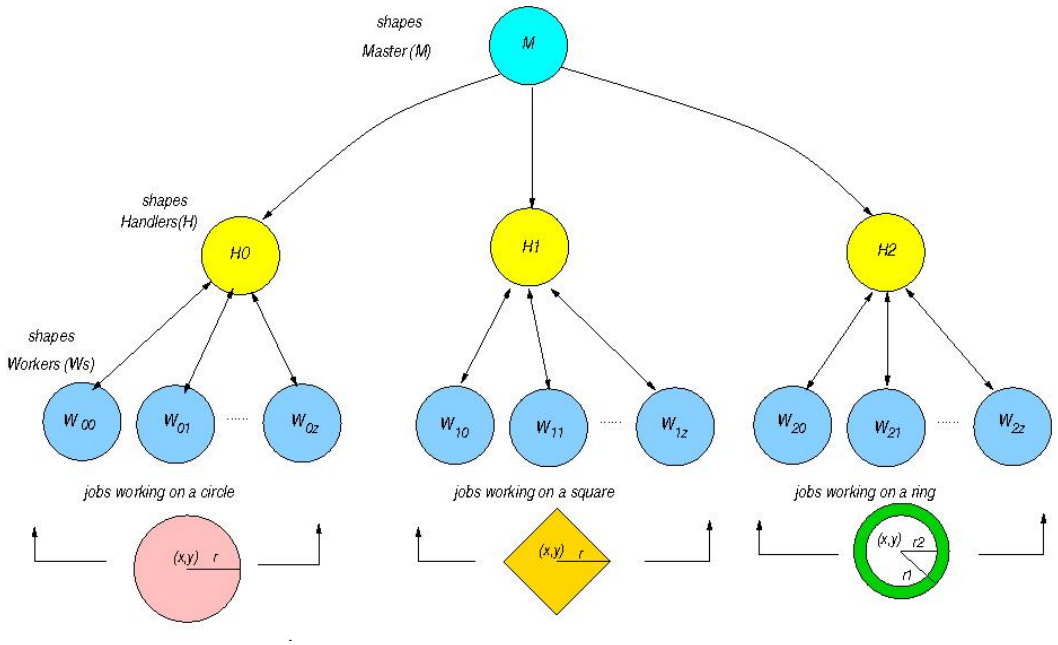
Παρομοίως μια κλήση του utility ellipse μπορείε να είναι ως εξής:
prompt> ./ellipse -i InputBinaryFile -n 100 -o OutputFile -f 80 -a 5.0 2.0 3.5 1.0.
όπου διαβάζουμε 100 σημεία από τη θέση 80 του αρχείου εισόδου InputBinaryFile και βρίσκουμε πόσα ανήκουν στη έλλειψη που ορίζεται απο τις παραμέτρους 5.0 2.0 3.5 1.0 και τα εκτυπώνουμε στο αρχείο εξόδου OutputFile.

Γενική περιγραφή προγράμματος shapes:

Το πρόγραμμα shapes μαζί με όλους του κόμβους διεργασιών που δημιουργεί παρέχει μια ιεραρχία διεργασιών που συνεργατικά μπορεί να απαντήσει χωρικές επερωτήσεις. Η ιεραρχία που θα δημιουργήσετε μπορεί να εκτελεί πολλαπλά (διαφορετικά ή και μη) χωρικά ερωτήματά στα σημεία του αρχείου εισόδου, και τέλος να εκτυπώνει τα σημεία τα οποία τα ικανοποιούν με χρήση του προγράμματος gnuplot [6].

Η Ιεραρχία Διεργασιών:

Η ιεραρχία που θα δημιουργήσετε είναι τρι-επίπεδη και αποτελείται από τον shapes Master (M), τους shapes Handlers (H), και τέλος, από τους Workers (W). Το Σχήμα 6 παρουσιάζει ένα παράδειγμα της ιεραρχίας που πρέπει να δημιουργήσει το πρόγραμμά shapes. Ο αριθμός των εργατών w που κάθε handler δημιουργεί παραμένει σταθερός σε όλη την διάρκεια εκτέλεσης του shapes και ορίζεται μέσω μιας παραμέτρου γραμμής εντολών. Στο παράδειγμα του Σχήματος 6 κάθε H που επιβλέπει ενός είδους επερώτηση, έχει δημιουργήσει και 'καθοδηγεί' z διαδικασίες-παιδιά (w=z) για να *ψάξουν ταυτόχρονα* τα δεδομένα του αρχείου δεδομένων. Ο handler H0



Σχήμα 6: Παράδειγμα ιεραρχίας διεργασιών με w=z για κάθε H

χρησιμοποιεί τους υποκείμενους z εργάτες ώστε οι τελευταίου όλοι ταυτόχρονα να βρουν τα σημεία που ανήκουν σε μια περιοχή κύκλου. Παρομοίως, ο H1 συντονίζει z εργάτες για να βρουν σημεία σε τετράγωνο και ο H2 απασχολεί z εργάτες για να βρουν σημεία σε ένα δακτύλιο. Οι επικοινωνία μεταξύ των handlers και των εργατών όσον αφορά στα αποτελέσματα γίνεται με named pipes. Ο κάθε handler 'συγκεντρώνει' τα επιμέρους αποτελέσματα που ο κάθε εργάτης δημιουργεί, και τα αποθηκεύει σε ένα προσωρινό αρχείο που τελικά γίνεται

[6] `http://gnuplot.sourceforge.net/`

διαθέσιμο στο master για την δημιουργία ενός γραφικού συνολικού αποτελέσματος.

Ο Ρόλος των Διεργασιών σε κάθε Επίπεδο:

Σε αυτό το σημείο περιγράφουμε το συγκεκριμένο ρόλο που κάθε διαδικασία παίζει στην ιεραρχία όπως και τις σχετικές διεπαφές.

• shapes Master (M):
Το αρχικό σας πρόγραμμα είναι αυτό το επίπεδο και εδώ η κλήση του εκτελέσιμου γίνεται με τα τα εξής ορίσματα:
`prompt> ./shapes -i InputBinaryFile -w WorkersCount -d TempDir`
όπου:

- `shapes` είναι το εκτελέσιμο της εφαρμογής που δημιουργεί την ιεραρχία διεργασιών,
- `InputBinaryFile` είναι το δυαδικό αρχείο με τα δεδομένα εισόδου γραμμικοποιημένα (serialized) σε μορφή float, float, ... (χωρίς κενό ή κόμμα μεταξύ τους),
- WorkersCount είναι το πλήθος των κόμβων w workers που θα πρέπει να δημιουργηθούν από κάθε shape Handler (H) κόμβο,
- TempDir είναι ένας κατάλογος τον οποίο θα χρησιμοποιήσει το πρόγραμμά σας για την προσωρινή αποθήκευση ενδιάμεσων αρχείων (αν δεν υπάρχει, θα πρέπει να δημιουργηθεί).

Οι σημαίες `-i/ -w/ -d` μπορούν να χρησιμοποιηθούν με οποιαδήποτε σειρά στην γραμμή εκτέλεσης.

Όταν ο master εκτελεσθεί, θα πρέπει αρχικά να διαπιστώσει το αριθμό των δεδομένων του αρχείου. Ο master, εκτός από το πλήθος των σημείων, θα χρειαστεί να υπολογίσει για τον κάθε worker την θέση (offset) από την οποία θα ξεκινήσει το διάβασμα του αρχείου. Οι εργάτες θα έχουν διαφορετικά Offset για το αρχείο εισόδου αλλά είναι και πολύ πιθανόν ο τελευταίος λογικά να έχει και μικρότερο αριθμό σημείων να επεξεργαστεί (PointsToReadCount).

Η διεργασία M υποστηρίζει ένα Command Line Interface (CLI) μέσω του οποίου, ο χρήστης θα μπορεί να εισάγει συγκεκριμένα utilities (ένα ή πιο πολλά) που θα πρέπει να εκτελεστούν καθώς και τα σχετικά ορίσματα τους. Τα τελευταία είναι διαθέσιμα από την τιμή/ές της παραμέτρου `-a`. Όταν ένας χρήστης βρεθεί στο CLI είτε μπορεί να κάνει `exit()` και να ολοκληρωθεί η εκτέλεση του `shapes` είτε να δεχτεί εντολές με την παρακάτω μορφή:
⟶ `shape1 arg1 arg2 ...argN;`
με το `shape1` να είναι το πρόγραμμα χρήσης της επιλογής μας, `arg1 arg2 ...argN` είναι οι απαιτούμενες παράμετροι, και το ερωτηματικό ; να υποδηλώνει το πέρας της εντολής.

Το πρόγραμμα σας θα πρέπει επίσης, να υποστηρίζει ομαδική εκτέλεση πολλαπλών utilities, διαχωρίζοντας τις διαφορετικές εντολές με κόμματα ακολουθώντας την εξής μορφή:
⟶ `shape1 arg11 arg12 ...arg1N, shape2 arg21 arg22 ...arg2M, .....  ;`

Για παράδειγμα, για να εκτελέσουμε ομαδική επερώτηση ενός 'κόκκινου' κύκλου, ενός 'πορτοκαλί' τετραγώνου και ενός 'πράσινου' δακτυλίου, θα εισάγουμε το εξής:
⟶ `circle 3.0 2.2 5.0 red, square 3.0 3.8 2.5 orange, ring 5.0 5.0 4.0 4.5 green;`
Με την παραπάνω εντολή, ο master θα δημιουργήσει τρεις handlers εκ των οποίων ο κάθε ένας είναι υπεύθυνος να εκτελέσει την κάθε μία από τις τρεις αυτές utility με την βοήθεια w εργατών.

Κάθε φορά που το CLI διαβάζει μια γραμμή εισόδου, θα μπορεί να αναγνωρίσει το πλήθος των διαφορετικών

εντολών που θα πρέπει να εκτελεσθούν, το εκάστοτε πρόγραμμα χρήσης και τα επιμέρους ορίσματα τους. Έπειτα, θα χρειαστεί να δημιουργήσει τους handlers που χρειάζονται τις εξής παραμέτρους:

- το πλήθος των `workers` που θα δημιουργηθούν
- ποιο πρόγραμμα χρήσης (utility) θα εκτελεσθεί
- τις επιμέρους παραμέτρους του προγράμματος χρήσης (χωρίς όμως την πληροφορία του χρώματος). Πιο συγκεκριμένα, οι παράμετροι αυτοί είναι οι παρακάτω: `InputBinaryFile, OutputFile, UtilityArgs, Offset` και `PointsToReadCount`.

• shapes Handler (H):
Έπειτα από την προετοιμασία των παραμέτρων, ο handler θα δημιουργήσει w εργάτες. Οι τελευταίοι με την σειρά τους θα εκτελέσουν το utility που τους αναλογεί. Για καθέναν από τους workers, ο handler θα δημιουργήσει ένα `FIFO` (named pipe), του οποίου το όνομα θα περαστεί σαν παράμετρος αρχείου εξόδου στο εκάστοτε utility. Το αρχείο (δηλ. το named pipe) θα δημιουργηθεί στον προσωρινό κατάλογο (TempDir) που έχει δοθεί στην γραμμή εντολής του shapes. Επίσης, το όνομα του `FIFO`, θα ακολουθεί την σύμβαση `PIDparent_wCount.fifo`. Για παράδειγμα, με $w = 2$, ο handler με $PID = 5000$, θα δημιουργήσει δύο `FIFO`ς με ονόματα: `5000_w0.fifo, 5000_w1.fifo`. Στην συνέχεια, θα περάσει σαν παράμετρο αρχείου-εξόδου το όνομα του κάθε named pipe (δηλ. το `OutputFile`) στον αντίστοιχο εργάτη που εκτελεί το utility.

Ο handler θα αναμένει το τερματισμό των εργατών ώστε να μπορέσει να συνθέσει ένα *μοναδικό προσωρινό αρχείο* (που βρίσκεται στο temporary directory) με αποδοτικό τρόπο (δηλαδή χωρίς να χρειάζεται να περιμένει ο handler όλους τους εργάτες του να τερματίσουν πριν αρχίσει την σύνθεση του αρχείου αποτελέσματος). Το όνομα του προσωρινού αρχείου αποτελέσματος θα ακολουθεί την συμβατική μορφή `SHPID.out`, όπου το `SHPID` είναι το *PID* του handler. Για παράδειγμα ο handler με $PID = 5000$, θα δημιουργήσει αρχείο εξόδου `5000.out`.

• Παρουσίαση Τελικού Αποτελέσματος από το shapes Master (M):
Αφότου αναμείνει τον τερματισμό όλων των handlers, ο master έχει πρόσβαση στο αποτέλεσμα της εκτέλεσης τους μέσω των προσωρινών τους αρχείων. Ο master γνωρίζει τα *PID*s των handlers, και κατ΄ επέκταση τα ονόματα όλων των ενδιαμέσων αρχείων αποτελεσμάτων που πρέπει να συνθέσει.

Ο master θα δημιουργήσει ένα gnuplot script αρχείο στον προσωρινό κατάλογο. Το όνομα του αρχείου θα ακολουθεί την σύμβαση `CommandCount_script.gnuplot` όπου το `CommandCount` είναι ο αύξων αριθμός της εκάστοτε εντολής που έχει δεχτεί το `CLI`. Το `gnuplot` θα οπτικοποιήσει τα αποτελέσματα των επερωτήσεων παίρνοντας σαν είσοδο το `script`. Το παράρτημα παρέχει βασικές πληροφορίες για την χρήση του `gnuplot` στην εν λόγω περίπτωση.

Μόλις ολοκληρωθεί η οπτικοποίηση των αποτελεσμάτων με το `gnuplot`, ο master θα είναι σε θέση πάλι να δεχτεί νέες εντολές ή να τερματίσει. Στο τέλος εκτέλεσης του `shapes`, τα περιεχόμενα του προσωρινού καταλόγου θα διαγράφονται.

Διαδικαστικά:

- Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις, αξιολόγηση, βαθμολόγηση κλπ.) ο κ. Δημήτρης Μούρης `sdi1200114+AT-di`, και ο κ. Χρήστος Ασλάνογλου `caslanoglou+AT-di`.
- Το πρόγραμμά σας θα πρέπει να γραφτεί σε `C` (ή `C++` αν θέλετε αλλά χωρίς την χρήση `STL/Templates`) και να τρέχει στις μηχανές `Linux workstations` του τμήματος.
- Η χρήση του `Makefile` είναι *επιτακτική!*
- Παρακολουθείτε την ιστοσελίδα του μαθήματος `http://www.di.uoa.gr/~ad/k22/` για επιπρόσθετες

ανακοινώσεις αλλά και την ηλεκτρονική-λίστα (η-λίστα) του μαθήματος στο URL `https://piazza.com/uoa.gr/fall2016/k22/home`

- Μέσα από την σελίδα `https://piazza.com/uoa.gr/fall2016/k22/home` θα μπορείτε να κάνετε ερωτήσεις και να δείτε απαντήσεις ή/και διευκρινήσεις που δίνονται σχετικά με την άσκηση (όπως έγινε με την πρώτη άσκηση).

Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (1-2 σελίδες σε ASCII κειμένου είναι αρκετές).
2. Οπωσδήποτε ένα `Makefile` (που να μπορεί να χρησιμοποιηθεί για να γίνει αυτόματα το compile του προγράμματος σας). Πιο πολλές λεπτομέρειες για το (`Makefile`) και πως αυτό δημιουργείται δίνονται στην ιστοσελίδα του μαθήματος.
3. Ένα `tar-file` με όλη σας την δουλειά σε έναν κατάλογο πιθανώς να φέρει το όνομα σας και θα περιέχει όλη σας την δουλειά δηλ. `source files, header files, output files` (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

Άλλες Σημαντικές Παρατηρήσεις:

1. Οι εργασίες είναι **ατομικές**.
2. Το πρόγραμμα σας θα πρέπει να τρέχει στα `Linux` συστήματα του τμήματος αλλιώς **δεν μπορεί να βαθμολογηθεί**.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) είναι κάτι που **δεν επιτρέπεται** και δεν πρέπει να γίνει. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικά **απλά παίρνει μηδέν** στο μάθημα. Αυτό ισχύει για **όσους εμπλέκονται ανεξάρτητα** από το ποιος έδωσε/πήρε κλπ.
4. Υποβολές που **δεν χρησιμοποιούν** `Makefile` χάνουν αυτόματα 5% του βαθμού.
5. Σε καμιά περίπτωση τα *Windows δεν είναι επιλέξιμη* πλατφόρμα για την παρουσίαση αυτής της άσκησης.

Παράρτημα: Χρήση gnuplot για Εμφάνιση Σημείων

Το πρόγραμμά `shapes` θα δημιουργεί ένα αρχείο `CommandCount_script.gnuplot` για κάθε `CLI` εντολή που δίνεται για εκτέλεση (όπου `CommandCount` να είναι ο αύξων αριθμός της εκάστοτε εντολής του `CLI`) με την εξής επικεφαλίδα κάθε φορά:

```
set terminal png
set size ratio -1
set output "./CommandCount_image.png"
plot \
"tmp_dir/SHPID0.txt" notitle with points pointsize 0.5 linecolor rgb "green",\
"tmp_dir/SHPID1.txt" notitle with points pointsize 0.5 linecolor rgb "yellow",\
"tmp_dir/SHPID2.txt" notitle with points pointsize 0.5 linecolor rgb "blue"
```

Το παραπάνω αρχείο, θα αποτελέσει την είσοδο του `gnuplot`.

Το όνομα του αρχείου εικόνας που θα δημιουργηθεί είναι της μορφής `CommandCount_image.png`. Για παράδειγμα, αν το `gnuplot` αρχείο λέγεται `10_script.gnuplot`, τότε το όνομα της αντίστοιχης εικόνας θα είναι `10_image.png`.