1η Εργασία

Ονοματεπώνυμο: Ιωάννης Γρυπιώτης
Α.Μ.: 1115202100028

Πως να τρέξετε το πρόγραμμα.Δομή αρχείων.

- Όσο αφορά την δομή των αρχείων του "*project*".Τα έχω ξεχωρίσει σε φακέλους προκειμένου να μπορεί να γίνει πιο εύκολη και καθαρή η υλοποίηση της εργασίας.Παρακάτω αναφέρονται οι φάκελοι και την λογική πίσω από αυτούς.
  - Ο φάκελος /**inc** συμπεριλαμβάνει όλες τις δίκες μου βιβλιοθήκες **.h** που υλοποίησα και χρησιμοποίησα.
  - Ο φάκελος /**information** συμπεριλαμβάνει όλα τα **.txt**. Τόσο τα 3 **confige** όσο και το κειμενάκι που μας δώσατε προκείμενου να δούμε ότι η υλοποίηση μας είναι σωστή.
  - Ο φάκελος /**obj** συμπεριλαμβάνει όλα τα αρχεία από το **compilation** σε μορφή **.o** προκείμενου μετά να τρέξω το πρόγραμμα.
  - Ο φάκελος /**src** συμπεριλαμβάνει όλα τα αρχεία κώδικα **.c** τα οποία δεν περιέχουν καμιά **main()**
  - Τέλος έχω τα 2 αρχεία **main()** δηλαδή το **client.c** και το **server.c** τα οποία περιέχουν τα 2 κύρια προγράμματα της εργασίας μου.Εκτός από αυτά υπάρχει και ένα **makefile** που κάνει το **compilation** και την σύνθεση της εργασίας πολύ πιο απλή και εύκολη υπόθεση
- Όσο αφορά τον τρόπο με τον οποίο θα τρέξετε την εργασία έχω φτιάξει ένα **makefile** που διευκολύνει την όλη διαδικασία.Παρακάτω παρέχονται οι εντολές άλλα και το τι κάνουν.
  - **make**: Κάνει **compile** όλα τα αρχεία από τον φάκελο /**src** και δημιουργεί τα **.o** αρχεία μέσα στον φάκελο /**obj**.Ακόμα δημιουργεί τα εκτελέσιμα αρχεία **client** και **server**.
  - **./server <max client at the same time> <txt path,for the random lines> <confige.txt choise>**: Εκτέλεση του προγράμματος **server** με τα παραπάνω 3 ορίσματα.
  - Προσοχή για να λειτουργήσει σωστά το παρακάτω θα πρέπει να κάνετε πρώτα την εντολή **make**.
    make valgrind_server ARGS="<max client> <txt path> <confige choise>": Τρέχει την εφαρμογή μαζί με το **valgrind** προκείμενου να μπορούμε να δούμε αν έχουμε κάπου **memmory leaks or errors,etc**.
  - **make clean**: Διαγράφει όλα τα αρχεία στο φάκελο /**obj** μαζί με τα εκτελέσιμα **server**,**client**.

Μέσα στο αρχείο **struct.h** έχω φτιάξει 2 πολύ βασικά **structs** τα οποία στην ουσία αποτελούν ότι χρειάζομαι όσο άφορα κύριος την σωστή οργάνωση των δεδομένων μου για την υλοποίηση αρκετών κομματιών της εργασίας.

Παρακάτω θα αναφέρω για κάθε .h τα πιο σημαντικά κομμάτια μεμονωμένου κώδικα

list.h

Η βιβλιοθήκη αυτή είναι υπεύθυνη για την υλοποίηση μιας συνδεδεμένης λίστας.Έχω φτιάξει 2 πολύ σημαντικά με λίγα λόγια **structs**.

- Το Node,το οποίο το χρησιμοποιώ προκειμένου να φτιάξω το **Link list**
- Το List_MetaData,το οποίο περιέχει
  - 2 ακέραιους προκειμένου να ξέρω την ακριβείς χρονική στιγμή που θα πρέπει να εκτελέσω την 1η εντολή από το **confige**,(int Min_Time_Snapshot;) αλλά και τον συνολικό χρόνο που θα πρέπει να τρέχει το πρόγραμμα δηλαδή τον ακέραιο πριν το **EXIT**,(int Max_Time_Snapshot;)
  - 3 δείκτες σε Nodes προκειμένου να έχουμε κάποιες επιμέρους λειτουργικότητες.Δείκτη στο 1ο στοιχείο της λίστας (Nodes *start;) το οποίο με βοηθάει στο κάνω γρήγορη διαγραφή δεδομένων στην λίστα.Δείκτη στο τελευταίο στοιχείο της λίστας (Nodes *end;) προκειμένου να μπορώ να κάνω γρήγορη εκχώρηση καινούριων δεδομένων.Δείκτης για να κάνω itarate (Nodes *it;) τα δεδομένα της λίστας μου.

read_funs.h

Η βιβλιοθήκη αυτή είναι υπεύθυνη για την υλοποίηση συναρτήσεων για διάβασμα δεδομένων και αποθήκευση τους σε δικές μου δομές από αρχεία ,(**confige file**) και (αρχείο με γραμμές για random διαβασμα).Επίσης έχουν δηλωθεί με (# define) τα 3 διαφορετικά μονοπάτια για τα 3 διαφορετικά **confige file** με ονομασίες ανάλογες με το μέγεθος τους.

- #define READ_BUFFER_SIZE 8.Το οποίο το χρησιμοποιώ σαν μέγεθος για το buffer στο αρχείο που διαβάζω της εντολές.
- #define DATA_BUFFER_SIZE 1024.Το οποίο είναι το μέγεθος του buffer προκείμενου να μπορέσω να διαβάσω μια ολόκληρη γραμμή από το αρχείο στο οποίο πρέπει να επιλέγω συνέχεια μια τυχαία γραμμή.Προσοχή στην περίπτωση που η κάθε γραμμή του αρχείου αποτελείτε από παραπάνω από 1023 bytes θα υπάρχει πρόβλημα.

sema.h

Εδώ πέρα έχω την υλοποίηση των ονοματισμένων **σεμαφόρων** προκειμένου να γίνει σωστά ένα τεράστιο μέρος του σωστού συγχρονισμού των διεργασιών μεταξύ **clients,server**.Γενικά έχω φτιάξει το Sem_Infos το οποίο περιέχει στην ουσία το **label** (int Pr_label;),τον ακέραιο,του **client** μαζί με το αντίστοιχο όνομα του σεμαφόρου (char Name[SEM_NAME_LEN];) που του αντιστοιχεί όπως και με έναν δείκτη στον σεμαφόρο (sem_t *ID;). Επομένως το όνομα παραμένει σταθερό,όπως και η τιμή του σεμαφόρου που παίρνει τιμές μόνο (0-1),ενώ αντίθετα αυτό που αλλάζει συνεχώς κατά την εκτέλεση του προγράμματος είναι το ποιο από όλα τα **client** στην ουσία θα πάρει ποιον ονοματισμένο σεμαφόρο.
Με λίγα λόγια το Sem_Infos μου χρησιμεύει προκειμένου γνωρίζοντας ποιος είναι ο **client** ο οποίος πρέπει να διαβάσει από την ενδιάμεση μνήμη,εγώ σαν **server** να μπορώ να κάνω **up** τον αντίστοιχο σεμαφόρο του.Ακόμα χρησιμεύει στο να ξέρει ο κάθε **client** ποιος ονοματισμένος σεμαφόρος του αντιστοιχεί.

- #define START_SEM_NAME "A";.Το όνομα του 1ου σεμαφόρου **client**.Μετά πάω διαδοχικά και με βάση το όρισμα που μου δίνετε για μέγιστο πλίθος **alive client** ,την ίδια στιγμή ,πάω και κάνω **generate** τους υπόλοιπους.Επομένος στην περίπτωση που μου ζήτησε παραπάνω από 26 ταυτόχρονους **alive client** (όσα και τα γράμματα του αγγλικού αλφαβήτου) κατα πάσα πιθανότητα το πρόγραμμα θα εκτελεστεί με έναν μη ντετερμινιστικό τρόπο.

- #define NO_LABEL 0;.Προκειμένου να ξέρω αν κάποιος συγκεκριμένως ονοματισμένος σεμαφόρος χρησιμοποιείτε από κάποιον **client**.

- #define SEM_INIT_VALUE 0;.Η αρχική τιμή που παίρνουν οι σεμαφόροι των **clients**.

server_fun.h

Η βιβλιοθήκη είναι υπεύθυνη για την υλοποίηση βοηθητικών συναρτήσεων που χρησιμοποιούνται στο κύριο πρόγραμμα προκειμένου να υπάρξει καλύτερος έλεγχος της όλης εκτέλεσης του προγράμματος αλλά και διευκόλυνση κατά την υλοποίηση της εργασίας.

shmemmo.h

Η βιβλιοθήκη είναι υπεύθυνη για την σωστή δημιουργία και διαχείριση της ενδιάμεσης μνήμης.Στην ουσία είναι ένα **wraper** γύρω από **System V shared memmory** συναρτήσεις.Ακόμα έχω φτιάξει το Shared_Block το οποίο στην ουσία μας δείχνει το τι βρίσκετε μέσα στην ενδιάμεση μνήμη.

- char buff[SIZE];.Η γραμμή που διαβάζει ο **server** από το αρχείο με τις γραμμές και που πρέπει μετά να το διαβάσει και να το εκτύπωση ο **client**.

- bool Terminate;.Ένα **flag** που υποδηλώνει στον **client** αν πρέπει να τερματίσει.

- int End_Time;.Ο χρόνος στον οποίο δόθηκε η εντολή να τερματιστεί ένας **client**,προκειμένου ο ίδιος να υπολογίσει τον συνολικό χρόνο που αυτός ήταν **allive**.

Τα 2 κυρία προγράμματα και οι λογικές μου

client.c

Τα ορίσματα που παίρνει από την γραμμή εντολών κατά την εκκίνηση του με την σειρά
./client <client semaphore name> <server semaphore name> <arrival time> <client label>

- Το όνομα που έχει ο σεμαφόρος που αντιστοιχεί σε αυτόν.Σημαντικό προκειμένου να υπάρχει σωστός συγχρονισμός μεταξύ **client-server**.

- Το όνομα που έχει ο σεμαφόρος του **server**.Σημαντικό προκειμένου να υπάρχει σωστός συγχρονισμός μεταξύ **client-server**.

- Η χρονική στιγμή στην οποία ξεκίνησε ο **client**.Σημαντικό προκειμένου να ξέρω το χρονικό διάστημα στο οποίο ήταν **allive**.

- Το **label** που έχει.Μόνο για μια πιο ωραία και πλήρης σε πληροφορίες εκτύπωση.

Βήματα και σημαντικά σημεία.

- Μετατρέπω σε ακέραιους το **label** και το **arrival time** προκειμένου μετά να μπορώ να κάνω πράξεις.
- Ανοίγω τους σεμαφόρους που αντιστοιχούν στον συγκεκριμένο **client** αλλά και στον **server**.
- Κάνω **attach** το **shared memmory** στον **client**.
- Εκτυπώνω ένα μήνυμα στο **terminal** προκειμένου να ξέρω πότε και ποιος **client** έγινε **SPAWN**.
- Κάνω έναν ατέρμονα βρόχο.Μέσα σε αυτόν κάνω τα εξής.
    - Κατεβάζω τον σεμαφόρο του **client**.Κάθε φορά που θα γίνει αυτό θα μπλοκαριστεί ο **client** μιας και ο σεμαφόρος είναι αρχικοποιημένος στο 0 πιο πριν απο τον **server**.Και θα ξεμπλοκαριστεί μόνο όταν του πει ο **server**.
    - Όταν θα αποκτήσει πρόσβαση στην διαμοιραζόμενη μνήμη θα είναι μόνος του και θα διαβάσει αν πρέπει να τερματίσει.Αν ναι τότε τερματίζει αλλιώς διαβάζει την γραμμή που πρέπει να εκτυπώσει και την εκτυπώνει,και ανεβάζει τον σεμαφόρο του **server** προκειμένου να συνεχίσει τις εργασίες που πρέπει να κάνει.

  Τα παραπάνω με του σεμαφόρους γίνονται προκειμένου να διασφαλίσουμε 2 πραγματάκια.

    - Το γεγονός ότι μόνο 1 **process** έχει κάθε χρονική στιγμή πρόσβαση στην διαμοιραζόμενη μνήμη.
    - Προκειμένου να δώσουμε τον χρόνο που χρειάζεται στον **client** για να κάνει τις δουλείες που του έχουμε βάλει να κάνει.
- Όταν έρθει η στιγμή να τερματίσει εκτυπώνω ένα μήνυμα **TERMINATE** μαζί με επιμέρους πληροφορίες όπως **label**,χρόνος που ήταν ζωντανός,πόσες φόρες διάβασε από την διαμοιραζόμενη μνήμη.Ακόμα κλείνω τους 2 σεμαφόρους που άνοιξα στην αρχή και κάνω **detach** την διαμοιραζόμενη μνήμη προκειμένου να μπορεί να διαγραφτεί μετέπειτα από τον **server**.

server.c

Τα ορίσματα που παίρνει από την γραμμή εντολών κατά την εκκίνηση του με την σειρά **./server <max client at the same time> <txt path,for the random lines> <confige.txt choise>**.Τα πρώτα 3 παρατέθηκαν και πιο πάνω.Όσο αφορά το **<confige.txt choise>** έχουμε τα εξής.

- 1: Το μικρο αρχείο με 3 **labels** και 100 επαναλήψεις
- 2: Το μεσαίο αρχείο με 3 **labels** και 1000 επαναλήψεις
- 3: Το μεγάλο αρχείο με 10 **labels** και 10000 επαναλήψεις

Αν βάλετε οτιδήποτε άλω θα σας βγάλει μήνυμα λάθους.

Βήματα και σημαντικά σημεία.

- Ελέγχω για το αν μου έδωσαν λιγότερα ή περισσότερα ορίσματα από την γραμμή εντολών.Αν ναι τερματισμός.
- Διάβασμα για το πόσες γραμμές έχω από το αρχείο με τα δεδομένα (όχι το **config**),δέσμευση ενός πίνακα όσες είναι και οι γραμμές και μετά αποθήκευση των γραμμών στο πινακάκι.

- Εύρεση του μονοπατιού **config** σύμφωνα με το τι έδωσε ο χρήστης σαν όρισμα.
- Αρχικοποίηση μιας λίστας και διάβασμα των εντολών από το αρχείο **config**,ενώ ταυτόχρονα αποθηκεύω τις εντολές στην λίστα.Αυτό το κάνω προκειμένου να ξέρω πόσες πρέπει να είναι οι συνολικές επαναλήψεις του προγράμματος.
- Δημιουργία της διαμοιραζόμενης μνήμης και **attach** στον **server**.
- Δημιουργία του σεμαφόρου του **server**.
- Δημιουργία ενός πίνακα τύπου (Sem_Infos) σε μέγεθος όσο και του 1ου ορίσματος που μου δίνει ο χρήστης στον **server** κατά την εκκίνηση της εργασίας.
- Έξαγωγή΄ της πρώτης εντολής από την λίστα.Και ορισμός ενός **flag** και αρχικοποίηση του προκειμένου να ξέρω αν είναι η κατάλληλη χρονική στιγμή να εκτελέσω κάποια εντολή όπως **SPWAN OR TERMINATE**.
- Επαναλήψεις αρχίζοντας την στιγμή που έρχεται η πρώτη εντολή μέχρις ότου την τελική τιμή που γράφει το **config** πριν το **EXIT**.
    - Σε περίπτωση που έχω κάποια εντολή μέσα στην λίστα και στην περίπτωση που δεν έχω στα ΄χέρια΄ κάποια εντολή προς εκτέλεση (το δεύτερο θα μπορεί να είναι αληθής διότι δεν έχει φτάσει ακόμα η κατάλληλη χρονική στιγμή προκειμένου να την εκτελέσω).
    - Στην περίπτωση που έχω κάποια εντολή για εκτέλεση και στην περίπτωση που είναι η κατάλληλη χρονική στιγμή να την εκτελέσω,βάζω το **flag** του τερματισμού σε **true** σε περίπτωση που πρέπει να τερματίσω ένα **client** και μετά την αλλάξω στην αρχική τιμή που είχε.
    - Πάω να εκτελέσω κάποιο **SPAWN** ή **TERMINATE**.
      Σε περίπτωση **SPAWN**
        - Ελέγχω αν μετά το **SPAWN** έχω ξεπεράσει το μέγιστο πλίθος από **clients**.Αν ναι εκτυπώνω ένα μήνυμα και τελειώνω την **for**.
        - Βρίσκω την 1η θέση που δεν είναι κατειλημμένη από κάποιο **process** στο πινακάκι από Sem_Infos και την θέση αυτήν την δίνω στο καινούριο **client** που έχω σκοπώ να κάνω **SPAWN**.
        - Μέτα πάω και κάνω **frok** και **exec** προκειμένου να φτιάξω τον **client**

      Σε περίπτωση **TERMINATE**
        - Πάω να βρω μήπως το **client** που πρέπει να κάνω **TERMINATE** δεν έχει γίνει ποτέ **SPAWN**.Αν ισχύει αυτό τότε απλά εκτυπώνω ένα μήνυμα και συνεχίζω κανονικά στην **for**.
        - Αν τελικά πρέπει να τερματίσω τον **client** τότε πάω και κάνω **up** τον αντίστοιχο σεμαφόρο του **client** προκειμένου να τον ξεμπλοκάρω.
        - Περιμένω να τερματίσει το **client** προκειμένου να πάω να τον μαζέψω έτσι ώστε να μην προκύψουν **zombie processes**.
    - Αν δεν έχω κάποιο **client allive** για να του δώσω κάποια δουλειά τότε απλά πάω στην επόμενη επανάληψη του **for**.
    - Πάω και βρίσκω μια τυχαία γραμμή από το αρχείο.
    - Πάω και διαλέγω έναν από τους τυχαίους **allive client** και του κάνω **up** τον σεμαφόρο.
    - Κάνω **wait** προκειμένου να περιμένω να τερματίσει ο **client** έτσι ώστε να μην μπουν ταυτόχρονα πάνω από 1 **process** στην διαμοιραζόμενη μνήμη.

- Πάω και τερματίζω όλους τους **clients** οι οποίοι δεν έχουν τερματιστεί ακόμα.Είτε επειδή τερματίστηκε βίαια το πρόγραμμα μου είτε επειδή τελείωσαν οι επαναλήψεις και δεν τους δόθηκε ποτέ εντολή τερματισμού μέσω του **config**.
- Πάω και αποδεσμεύω την μνήμη από την **Link List**.
- Πάω και αποδεσμεύω τους σεμαφόρους (**close,unlinck**) στο πινακάκι με τύπο Sem_Infos.
- Πάω και κάνω **close,unlinck** τους σεμαφόρους του **server**.
- Πάω και κάνω **dettach,shmctl** την διαμοιραζόμενη μνήμη.
- Απελευθερώνω την μνήμη που δέσμευσα δυναμικά για το πινακάκι που κρατούσε τις γραμμές που θα διαβάζω τυχαία.