ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ 2021-22

Εγκατάσταση και μεταγλώττιση προγράμματος (ΣΗΜΑΝΤΙΚΟ):

Μέσα στους φακέλους της εργασίας υπάρχει ένας φάκελος που λέγεται 'txt_files' και εκεί πρέπει να μπουν τα όποια αρχεία κειμένου για να τα ελεγχθεί το πρόγραμμα. Αν αποφασίσετε να βάλετε τα αρχεία κειμένου κάπου αλλού θα πρέπει να αλλαχθεί το path στην γραμμή 42 του 'server.c'. Το εκτελέσιμο πρόγραμμα είναι αυτό του server και θα μπορείτε να το τρέξετε γράφοντας './server arg1 arg2 … '. Επίσης θα πρέπει υποχρεωτικά το πρώτο όρισμα του προγράμματος να είναι το όνομα του αρχείου txt το δεύτερο όρισμα ο αριθμός των παιδιών και το τρίτο όρισμα ο αριθμός των δοσοληψιών (πχ. ./server test.txt 10 5), αν κάτι από αυτά δεν ισχύει τότε το πρόγραμμα δεν τρέχει γιατί έχει argument checking. Το αρχείο πρέπει να απαραίτητα να δοθεί μέσω του ονόματός του και σαν command line argument και θα ανοίξει μέσα στο πρόγραμμα κανονικά. Για την μεταγλώττιση γράφετε απλά 'make' στο terminal και θα φτιαχτεί μόνο του το εκτελέσιμο του server και όλα τα object files θα είναι μέσα σε έναν φάκελο που θα φτιαχτεί και θα ονομάζεται 'build'.

Σημείωση: Μέσα στον φάκελο txt_files υπάρχει ήδη ένα δικό μου αρχείο test.txt το οποίο έχει 90 γραμμές των περίπου 70 χαρακτήρων. Άρα με τις εντολές 'make' και μετά ' ./server test.txt 10 5 ' μπορείτε κανονικά να τρέξετε το πρόγραμμα.

Γενικές πληροφορίες για την εργασία:

- Ο κώδικας του server(parent) είναι στο server.c και του παιδιού στο client.c και καλείται μέσω της exec.
- Για το πρόγραμμα χρησιμοποιήθηκαν posix semaphores οι οποίοι είναι named οπότε δεν χρειάστηκε να αποθηκεύονται στην shared memory.
- Για την λειτουργικότητα του προγράμματος χρησιμοποιήθηκαν 3 semaphores σύνολο. Οι δύο είναι για την επικοινωνία server-client (producer-consumer) και ο τρίτος είναι για τον συγχρονισμό των παιδιών.
- Στο project υπάρχει επίσης και το αρχείο utilities.c το οποίο περιέχει κάποιες βοηθητικές συναρτήσεις οι οποίες:
  - Get_number_lines: μετράει τις γραμμές ενός αρχείου
  - Get_specific_line: φέρνει μία συγκεκριμένη γραμμή
  - Is_positive_number: βλέπει αν ένας αριθμός είναι θετικός και διάφορος του μηδέν (χρησιμοποιείται για το argument checking)
  - Number_to_str: Μετατρέπει έναν αριθμό σε char*
  - Get_sem_value: Φέρνει την current τιμή ενός semaphore (χρησιμοποιήθηκε στο debugging)
  - Print_sem_value: Εκτυπώνει την current τιμή ενός semaphore

Shared memory:

Η δομή της διαμοιραζόμενης μνήμης έχει μόνο έναν ακέραιο στον οποίο γράφουν τα παιδιά την γραμμή που ζητάνε και έναν πίνακα 101 χαρακτήρων στον οποίο γράφει ο γονέας τις γραμμές. Όπως προανέφερα οι semaphores που χρησιμοποιώ είναι posix άρα δεν χρειάστηκε να τους αποθηκεύσω εδώ.

‘Server.c ’:

Ο sever ξεκινάει ανοίγοντας το αρχείο txt και μετατρέποντας τα command line arguments σε μεταβλητές. Μετά δημιουργεί τους απαραίτητους 3 semaphores

και το κομμάτι της κύριας μνήμης που θα χρειαστούμε. Έπειτα δημιουργεί τα k παιδιά μέσω της execl και τους δίνει σαν arguments τον αριθμό των δοσοληψιών και τις συνολικές γραμμές του αρχείου όπως προβλέπει η εκφώνηση. Τώρα περνάμε στο
critical section:

Critical section:

Ο server κατεβάζει τον δικό του semaphore ο οποίο είχε αρχικοποιηθεί με την τιμή 0 οπότε θα πρέπει να τον ανεβάσει κάποιο παιδί για να γράψει την γραμμή στο shared memory. Μόλις ο semaphore αυτός ανέβει από κάποιο παιδί τότε γράφει την γραμμή στο shared memory και ανεβάζει τον πρώτο semaphore του παιδιού τον οποίο περίμενε το παιδί να ανέβει για να πάρει την γραμμή και να την εκτυπώσει (εξηγείται και παρακάτω).

Τέλος ο γονέας περιμένει όλα τα παιδιά να τερματίσουν μετά διαγράφει όλα τα semaphores και το shared memory segment και κάνει exit.

‘Client.c’ :

Ο client ξεκινάει μετατρέποντας τα ορίσματα που του έδωσε ο server σε μεταβλητές και μετά ανοίγει όλους τους semaphores και το shared memory segment (ο cli_sem είναι ο semaphore για την επικοινωνία με τον γονέα και ο cli_sem_2 είναι αυτός για την επικοινωνία μεταξύ clients). Εξήγηση του main loop των clients:

- Αρχικά περιμένουμε τον semaphore cli_sem_2 ο οποίος μας λέει ότι ένας άλλος client είναι στο critical section του οπότε πρέπει να περιμένουμε να ελευθερωθεί ο πόρος.
- Κατεβάζουμε τον cli_sem ο οποίος είναι ο client semaphore για την επικοινωνία με τον γονέα και ο οποίος είχε αρχικοποιηθεί με την τιμή 1 στην δημιουργία του οπότε το πρώτο παιδί τον βρίσκει με πόρο διαθέσιμο.
- Ζητάμε μια τυχαία γραμμή (χρησιμοποιώντας rand) γράφοντας στο shared memory και αρχίζουμε το χρονόμετρο

- Ανεβάζουμε τον serv_sem ο οποίος είναι ο semaphore του server και ο οποίος server τον περιμένει να ανέβει για να γράψει κάποια γραμμή στο shared memory.
- Μόλις ο γονέας τελειώσει μας ανεβάζει τον cli_sem τον οποίο εμείς περιμένουμε ξανά για να πάρουμε την γραμμή και να την εκτυπώσουμε
- Παίρνουμε εκτυπώνουμε την γραμμή που μας έδωσε ο server στο shared memory
- Ανεβάζουμε τον cli_sem για την επικοινωνία του επόμενου παιδιού με τον γονέα και ανεβάζουμε και το cli_sem_2 για να τρέξει κάποιο άλλο παιδί.
- Κοιμήζουμε τον current client με usleep(1) δηλαδή απειροελάχιστο χρόνο έτσι ώστε να μην περάσει στο επόμενο loop και κάποιος τυχαίος από τους άλλους clients να εκτελεστούν. Ουσιαστικά είναι ένας τρόπος για να επιλεγεί κάποιος άλλος τυχαίος client από το σύστημα.

Τέλος ο client κλείνει τους semaphores και κάνει exit εκτυπώνοντας κατάλληλο μήνυμα και τον μέσο χρόνο εκτέλεσης των δοσοληψιών του.

Σημείωση: Με το δικό μου text αρχείο επειδή μάλλον είναι πολύ μικρό το πρόγραμμα τρέχει ακαριαία και η clock δίνει ως μέσο χρόνο εκτέλεσης για το κάθε παιδί 0.0000 seconds.