ΧΡΗΣΤΟΣ-ΠΑΝΑΓΙΩΤΗΣ ΠΑΛΑΜΙΔΑΣ
1115201800140

Αρχεία:

1. server.c
2. client.c
3. destroy.c
4. memory.c
5. memory.h
6. text.txt & pdf επεξήγησης
7. makefile

Επεξήγηση αρχείων:

A. Το αρχείο server.c είναι στην πραγματικότητα το parent process. Στην αρχή του προγράμματος υπάρχουν κάποιες εντολές για error checking, εμφανίζοντας τα κατάλληλα μηνύματα λάθους στον χρήστη. Μέσω του server γίνεται η αρχικοποίηση της shared memory καθώς και των semaphores. Στο αρχείο αυτό, μιας και είναι το κύριο πρόγραμμα, καλείται η fork(). Αν η fork() επιστρέψει αρνητική τιμή (<0) τότε εμφανίζεται μήνυμα λάθους και επιστρέφεται η τιμή -1. Αν η fork() επιστρέψει τιμή 0, τότε εκτελείται το child process το οποίο καλεί για εκτέλεση το πρόγραμμα client (client.c), που επικοινωνεί μέσω των semaphores με το parent process. Αν η fork() επιστρέψει θετική τιμή (>0) τότε εκτελείται το parent process, το οποίο διαβάζει μέσω του shared memory το αίτημα του client και γράφει (στο shm) τη γραμμή που ζήτησε ο client. Τέλος το αρχείο server “μαζεύει” το id του παιδιού, “κλείνει” τους semaphores και κανει detach το shared memory. (περισσότερες πληροφορίες στην ‘Επεξήγηση της ροης του προγράμματος’)

B. Το αρχείο client.c, το οποίο όπως προαναφέρθηκε καλείται από το child process, κάνει αιτήματα προς το server. Στην αρχή, κάνει attach το ήδη υπάρχον shared memory καθώς και τους semaphores. Στην συνέχεια αφήνει το αίτημα (τη γραμμή που θα διαβάσει και γράψει ο server) στο shared memory, ο server κάνει τη δουλειά του και τέλος ο client διαβαζει και τυπώνει την κατάλληλη γραμμή. Επίσης το πρόγραμμα client, υπολογίζει και το χρόνο εκτέλεσης του.

C. Το αρχείο destroy.c, διαγράφει το shared memory που έχουμε δημιουργήσει μέσω του server. ΠΡΟΣΟΧΗ! Όπως έχει προαναφερθεί, ο server πριν το τελος της εκτέλεσης του κάνει detach τη διαμοιραζόμενη μνήμη, αλλά ΔΕΝ τη διαγράφει. Η διαγραφή επιτυγχάνεται με την εκτέλεση του destroy.

D. Τα αρχεία memory.c και memory.h αφορούν τη διαμοιραζόμενη μνήμη. Σε αυτά τα αρχεία περιέχονται οι συναρτήσεις: ‘get_shared_memory’ που παίρνει το κλειδί του shared memory, ‘attach_memory_block’ που κάνει attach της διαμοιραζόμενη μνήμη στο πρόγραμμα μας, ‘detach_memory_block’ που κάνει detach τη διαμοιραζόμενη μνήμη από το πρόγραμμα μας και ‘destroy_memory_block’, η οποία καλείται από το πρόγραμμα destroy.

E. Το αρχείο text.txt περιέχει 50 γραμμές με τυχαίο κείμενο. Οι γραμμές αυτές δεν είναι γραμμένες από εμένα αλλά από ένα Random Sentence Generator που βρήκα από το διαδίκτυο. (https://randomwordgenerator.com/sentence.php)

Επεξήγηση ροής προγράμματος:

Η εκτέλεση του προγράμματος ξεκινάει με τον server να ειναι το parent process. Μετά το error checking, το create/attach της διαμοιραζόμενης μνήμης και τη δημιουργία των semaphores, ξεκινάει ένα for loop τόσων επαναλήψεων, όσων και τα παιδιά που ζητάει ο χρήστης. Μέσα σε αυτό το loop, καλείται η fork(). Σκοπός είναι η δημιουργία όλων των παιδιών μαζί. Στη συνέχεια και βγαίνοντας από το loop, πλεον εκτελεί ο parent-server. Να σημειωθεί ότι κάθε child process, μόλις αρχίζει η εκτέλεση του client.c, κάνει sem_wait και περιμένει κάποιο σήμα από τον server για να ξεκινήσει. Ο server-parent λοιπόν κάνει sem_post ένα τυχαίο child-client και αμέσως μετά κάνει sem_wait() τον εαυτό του ώστε να περάσει η ροή στο παιδί. Εκείνο ξεκινάει να κάνει κάποιο request αφήνοντας έναν τυχαίο ακέραιο [1,2,...,50] στο shared memory που δηλώνει ποια γραμμή ζητάει. Στη συνέχεια το παιδί κάνει sem_post() τον server και sem_wait() τον εαυτό του. Ο server λοιπόν που παίρνει τη ροή του προγράμματος, διαβάζει από το shared memory το αίτημα, και γράφει μέσα εκεί την κατάλληλη γραμμή. Στη συνέχεια κάνει sem_post() τον client με τον οποίο ήδη επικοινωνούσε (!) και sem_wait() τον εαυτό του. Τελειώνοντας έτσι, ο client διαβάζει τη γραμμή που ζήτησε από το shared memory και ετοιμάζεται να κάνει το επόμενο. Επειδή όμως υπάρχουν πολλοί clients-children που περιμένουν (sem_wait), είναι τυχαίο ποιος από όλους θα κάνει το επόμενο αίτημα. Μετά από την επικοινωνία αυτή server και όλων των παιδιών, αφού αυτά ζητήσουν μια γραμμή και τη διαβάσουν, υπάρχει ένα for loop στον server-parent ο οποίος μαζεύει το id των παιδιών και εμφανίζει ότι ολοκλήρωσαν επιτυχώς. Τέλος κλείνει όλους τους semaphores και ΑΠΟΔΕΣΜΕΥΕΙ τη μνήμη (δεν την καταστρέφει όπως έχει ειπωθεί και από πάνω).

Εκτέλεση προγράμματος:

- Για το compile του προγράμματος αρκεί να γράψετε “make”.
- Για την εκτέλεση του προγράμματος:
    - ./server ./client 3 2
    - όπου ./server είναι το εκτελέσιμο, ./client είναι το πρόγραμμα που θα καλέσει το child process (αν κληθεί διαφορετικά εμφανίζεται μήνυμα λάθους και τερματίζει το parent process), ‘3’ είναι τα παιδια που θα δημιουργηθούν και ‘2’ είναι τα αιτήματα που θα κάνει το κάθε παιδί.
    - το ‘3’ και το ‘2’ που αφορούν το πλήθος τον παιδιών και των αιτημάτων τους είναι μεταβλητά, μπορείτε να δώσετε ό,τι τιμή θέλετε.
- Για την διαγραφή του shared memory αρκεί να γράψετε “./destroy”.
- Για την διαγραφή όλων των εκτελέσιμων αρκεί να γράψετε “make clean”.