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 πέρα από την εμφάνιση κάποιων μηνυμάτων, καλεί για εκτέλεση το πρόγραμμα client (client.c), το οποίο 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 οι οποίοι είναι 2, ένας για τον server και ένας για τον client, ξεκινάει ένα for loop τόσων επαναλήψεων, όσων και τα παιδιά που ζητάει ο χρήστης. Μέσα σε αυτό το loop, καλείται η fork(). Το parent process κάνει sem_wait στο semaphore του και περιμένει (οι semaphores αρχικοποιούνται με 0, οπότε με αυτο το wait δεν υπάρχουν διαθέσιμοι πόροι και άρα περιμένει μέχρις ότου να υπάρξει). Επομένως, η “ροη” περνάει στο child process, το οποίο με τη σειρά του, μέσω της execvp, καλεί τον client. Ο client ξεκινάει κάνοντας τις κατάλληλες αρχικοποιήσεις (shm,sem), αρχίζει να μετράει το χρόνο εκτέλεσης του και γράφει μέσα στο shared memory έναν τυχαίο αριθμό [1,2,...,50] που δηλώνει τη γραμμή που αιτείται από τον server. Στην συνέχεια ο client κάνει sem_post το semaphore του server και sem_wait τον δικό του. Επομένως η “ροή” περνάει στο parent process. Εκείνο διαβάζει τη διαμοιραζόμενη μνήμη, ανοίγει το .txt αρχείο, διαβάζει τη γραμμή που αιτήθηκε ο client και την τυπώνει στο shared memory (τους 100 πρώτους χαρακτήρες αν υπάρχουν). Στη συνέχεια κλείνει το .txt κάνει sem_post τον semaphore του client και sem_wait τον δικό του. Ο client διαβάζει την γραμμή και την τυπώνει στον χρήστη. Η διαδικασία των αιτημάτων, του γραψίματος και του διαβάσματος, επαναλαμβάνεται τόσες φορές, όσες ζητήσει ο χρήστης. Μετά από αυτές τις επαναλήψεις, ο client τυπώνει το χρόνο που έτρεξε και κάνει return. Το parent process δέχεται το id του child και το τυπώνει μαζί με ένα μήνυμα επιτυχούς εκτέλεσης. Αφού τελειώσουν όλα τα αιτήματα όλων των παιδιών, ο server 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”.