Εργασία 3

Αναστάσιος Τερζής - sdi2000189

Αρχείο mutual.h:

Το αρχείο αυτό αξιοποιείται από όλα τα πηγαία αρχεία .c και αφορά στον ορισμό structs και prototypes συναρτήσεων.

- **Request:** Αυτό το struct συνιστά την δομή ενός αιτήματος και είναι το object type της ουράς που θα χρειαστεί για την επικοινωνία file server - customer. Τα πεδία ενος Request περιέχουν τις πληροφορίες για τις γραμμές του επιθημητού αρχείου, καθώς και το shmTempId που αντιστοιχεί στην προσωρινή μνήμη που θα δημιουργηθεί, την οποία και θα προσαρτήσει το thread που θα αναλάβει να εξυπηρετήσει το Request.
- **SharedMemory:** Το struct αυτό αντιστοιχεί στην σταθερή διαμοιραζόμενη μνήμη μέσω τις οποίας επoικοινωνεί ο file server και οι customer διεργασίες, για την διακίνηση των Requests. Για σωστό συγχρονισμό αξιοποιούμε έναν semaphore χρήσης mutex.
  Ο πίνακας: "Request queue[QUEUE_SIZE]", υλοποιεί την κλασσική fifo δομή δεδομένων.
- **TempSharedMemory:** Αυτό το struct αντιστοιχεί στην προσωρινή διαμοιραζόμενη μνήμη την οποία δημιουργεί μια customer διεργασία, προκειμένου να επικοινωνήσει με ένα server thread. Οι δύο semaphores "dataReady", "dataEaten", αξιοποιούνται για τη σωστή ακολουθία του διαβάσματος-γραψίματος από τις δύο πλευρές. Ο πίνακας χαρακτήρων "char block[BLOCK_SIZE]", αναπαριστά το buffer που θα φιλοξενεί κάθε φορά ένα block/γραμμή του αρχείου.

Αρχείο fileserver.c:

Στο αρχείο αυτό βρίσκεται ο κώδικας της main συνάρτησης, καθώς και η λογική της πλευράς του file server. Αρχικά λαμβάνονται τα ορίσματα N, K, L, l από την γραμμή εντολών. Έπειτα δημιουργείται η διαμοιραζόμενη μνήμη shm, η οποία θα αξιοποιηθεί από τον file server και από τις διεργασίες customer, για την τοποθέτηση/λήψη Requests. Αφου δημιουργηθούν οι N διεργασίες, ο file server μπάινει στην βασική λούπα. Συνθήκη επανάληψης: Μέχρι να ικανοποιηθούν όλα τα requests (N*L). Σε κάθε επανάληψη ο file server αποκτά αποκλειστική πρόσβαση στην queue της κοινής μνήμης και παίρνει ένα request από αυτήν. Αμέσως δημιουργεί ένα thread στο οποίο περνάει το Request για να ξεκινήσει η εξυπηρέτηση. Μετά το πέρας της λούπας, η γονική αυτή διεργασία περιμένει την λήξη όλων των customer διεργασιών, κάνει join σε όλα τα threads και αποδεσμεύει τελικά όλα τα resourses που είχαν αξιοποιηθεί.

Αρχείο thread.c:

Αυτο το αρχείο περιέχει αποκλειστικά την ρουτίνα που εκτελεί κάθε thread που γεννιέται από τον file server. Ο σκοπός του είναι να εξυπηρετήσει ένα και μοναδικό request, το οποίο και λαμβάνει σαν όρισμα. Στο ξεκίνημα του thread, εντοπίζεται αμέσως το id της προσωρινής διαμοιραζόμενης μνήμης που έχει δημιουργηθεί από την customer διεργασία που περιμένει την εξυπηρέτηση (το id είναι αποθηκευμένο σε πεδίο του struct request). Με βάση το id προσαρτάται η εν λόγω προσωρινή μνήμη έτσι ώστε να εκκινήσει η επικοινωνία. Μόλις προσαρτηθεί η μνήμη, το thread ανοίγει το ζητούμενο αρχείο και ξεκινάει την λούπα για να αρχίσει να γράφει μια μια τις γραμμές στο διαμοιραζόμενο block. Οι δύο semaphores dataReady, dataEaten επιτρέπουν στο να μην γράφει το thread νέα γραμή πριν τη διαβάσει η customer διεργασία και ανάποδα.

Αρχείο customer.c:

Στο αρχείο αυτό βρίσκεται όλος ο κώδικας που αφορά μια customer διεργασία. Αρχικά η διεργασία δημιουργεί ένα log file αποκλειστικά δικό της, στο οποίο θα αποθηκευτούν τα αποτελέσματα της εξυπηρέτησης όλων των αιτημάτων καθώς και μερικά στατιστικά. Κάθε επανάληψη της κύριας λούπας αφορά την "εξέλιξη της ζωής" ενός request. Πρώτα από όλα διαμορφώνεται το request (με τυχαίο τρόπο), έπειτα δημιουργείται η προσωρινή διαμοιραζόμενη μνήμη (της οποίας το id αποθηκέυεται τώρα στο request.shmTempId). Το επόμενο βήμα είναι να αποκτηθεί αποκλειστική πρόσβαση στην queue ώστε να γραφεί το request σε αυτήν. Αφού ολοκληρωθεί η τοποθέτηση αυτή, η διεργασία μπαίνει στην εσωτερική λούπα όπου θα λαμβάνει σε κάθε επανάληψη μια γραμμή του αρχείου. Αντίστοιχη χρήση έχουν πάλι οι semaphores dataReady, dataEaten ώστε να ληφθούν σωστά όλες οι γραμμές. Κάθε φορά που λαμβάνεται μια γραμμή αυτή τυπώνεται στο log file ως μέρος του $< Content >$. Μόλις ολοκληρωθεί η λήψη όλων των γραμμών, διαγράφεται η προσωρινή μνήμη. Πριν περάσει σε επόμενη επανάληψη η διεργασία "περιμένει" (με χρήση usleep) τόσο χρόνο ώστε να προσομοιωθεί η ιδέα "εκθετικά κατανεμημένος χρόνος μεταξύ αιτημάτων¨. Μόλις γίνουν οι επαναλήψεις για τα L αιτήματα, η διεργασια γράφει τα τελευταία και συνολικά στατιστικά για τον εαυτό της, όπως: πλήθος γραμμών που λήφθηκαν συνολικά για τα L αιτήματα, μέσος χρόνος μεταξύ αιτημάτων και το πλήθος φορών επιλογής κάθε αρχείου.

Μεταγλώττιση/Εκτέλεση του προγράμματος:

- Η εργασία είναι υλοποιημένη πλήρως σε γλώσσα C.
  Υπάρχει αρχείο MakeFile για χάρη αυτοματοποίησης της μεταγλώττισης και της εκτέλεσης.

  Για να εκτελέσετε το πρόγραμμα τρέξτε στον αρχικό κατάλογο:
  make run.

- Το MakeFile έχει προκαθορισμένα ορίσματα εισόδου: **20 10 200 1**. Δηλαδή δημιουργέι 20 customer διεργασίες, ο fileserver παρέχει 10 διαφορετικά αρχεία (θα τα βρείτε στον κατάλογο "files"), κάθε customer θα εκτελέσει 200 αιτήματα, και η τιμή του λ για τον εκθετικό χρόνο είναι ίση με 1.

- Για να δοκιμάσετε διαφορετικές τιμές, τροποποίησε κατάλληλα το ARGS=... στο MakeFile, ή εκτελέστε: ./fileserver $< N > < K > < L > < l >$.

- Με την ολοκλήρωση μιας εκτέλεσης, θα έχουν δημιουργηθέι N log αρχεία, ένα για κάθε customer διεργασία. Εκεί βρίσκονται αναλυτικά οι πληροφορίες για κάθε ένα request, ενώ στο **τέλος του αρχείου**, υπάρχουν συγκεντρωμένα μερικά συνολικά στατιστικά.

- Αξιοποιήστε την εντολή "make clean" για να διαγραφούν εντελώς όλα τα log και object (.o) αρχεία που δημιουργήθηκαν μετά από μια εκτέλεση.