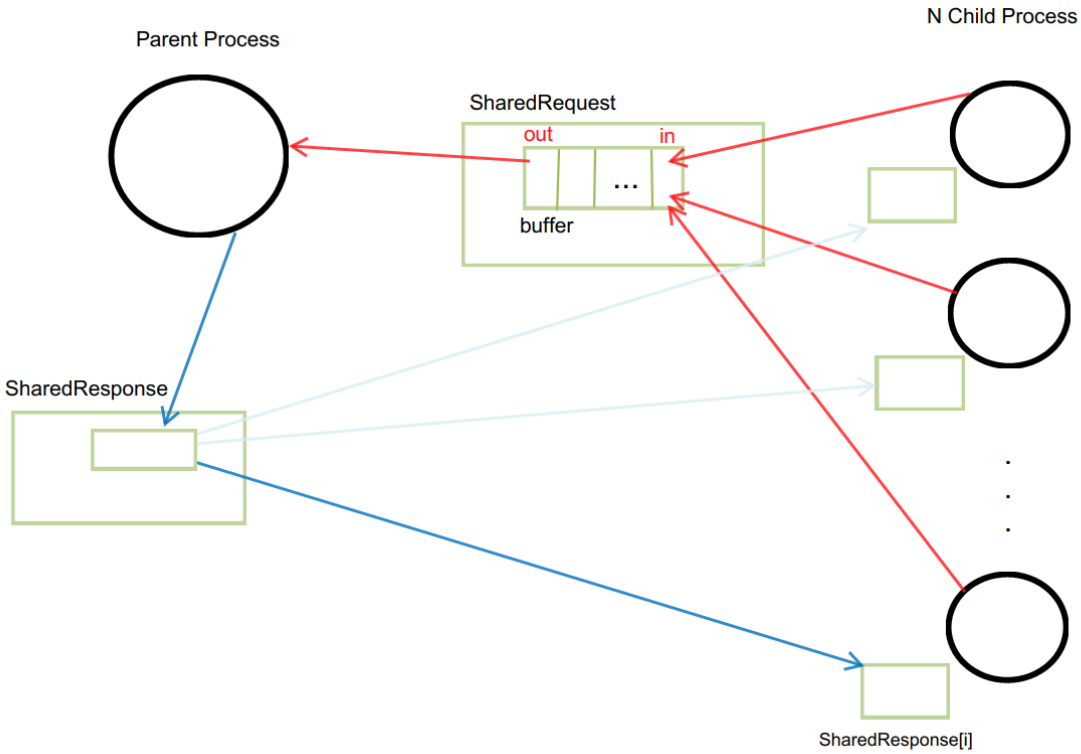
README

Ονοματεπώνυμο:Μαρίνα Μυλωνά

Sdi1900229

Εργασία 1

- **Περιγραφή μέσω σχήματος:**



Στο παραπάνω σχήμα απεικονίζεται η ιδέα με την οποία έχω υλοποιήσει την άσκηση. Αρχικά έχουμε Ν παιδιά τα οποία στέλνουν τα διάφορα request τους στο shared request όπου ο γονιός τα παραλαμβάνει με βάσει προτεραιότητας. Στη συνέχεια ο γονιός διαβάζει το συγκεκριμένο request(γραμμή) και το επιστρέφει στο συγκεκριμένο shared response του παιδιού. Κάθε φορά αλλάζει η σύνδεση του shared Response με το response του κάθε παιδιού ανάλογα με το ποιο response θα επιστραφεί.

- **Περιγραφή κώδικα:**

Έχουμε 3 διαφορετικά structs:

1. Το childRequest το οποίο έχει μέσα 3 ακεραίους που απεικονίζουν το αίτημα που θέλει να στείλει(id παιδιού, αριθμό segment και αριθμό γραμμής).
2. Το sharedRequest όπου έχουμε τους σημαφόρους που χρειαζόμαστε για την αποστολή και παραλαβή των αιτημάτων των παιδιών. Έχουμε ακόμα έναν σημαφόρου για την εκτύπωση. Επίσης έχουμε 2 ακέραιους, ο in είναι ο αριθμός που μας δείχνει σε ποια θέση του buffer θα μπει το request και ο out από ποια θέση θα πάρει ο γονιός το request που έχει σειρά. Τέλος έχουμε και ένα πίνακα από childRequest, τον buffer.
3. Το sharedResponse όπου έχουμε 2 σημαφόρους που χρειαζόμαστε για την αποστολή και παραλαβή των responses μεταξύ γονιού και παιδιών και έχουμε και την γραμμή όπου επιστρέφεται.

Στην main:

1. Παίρνουμε από τα εξωτερικά ορίσματα το όνομα του αρχείου, τον αριθμό των παιδιών, τον αριθμό των γραμμών ανά segment και τον αριθμό των request που κάνει το κάθε παιδί.
2. Δημιουργούμε το shared memory segment για τα request και τα response αντίστοιχα, αρχικοποιώντας τους σημαφόρους και τους ακεραίους.
3. Δεσμεύουμε χώρο για το segment όπου θα χρησιμοποιήσουμε για να φορτώσουμε όλο το segment.
4. Ανοίγουμε το αρχείο που θα χρειαστούμε και μετράμε όλες τις γραμμές του για να βρούμε σε πόσα segment θα χωριστεί το αρχείο. Αν η διαίρεση όλων των γραμμών/τις γραμμές ανά segment δεν βγαίνει ακέραιος αριθμός τότε προσθέτω ένα segment για τις extra γραμμές.
5. Έχω ένα for loop όπου δημιουργώ N παιδιά με fork().
6. Ανοίγω το αρχείο του γονιού.
7. Έχω ένα for loop όπου για κάθε request κάθε παιδιού κάνω την δουλειά για τον γονιό. Δηλαδή με βάσει το πρωτόκολλο producer-consumer ο γονιός παίρνει το request από τον buffer στη θέση out και μεταφέρει τον δείκτη αυτό στην επόμενη θέση. Αν βρίσκεται στην τελευταία θέση του buffer τότε επιστρέφει στην αρχή του buffer(κυκλικά). Μετά ελέγχω αν το segment που είναι φορτωμένο από προηγούμενη φορά είναι διαφορετικό από το segment που ζητάει το παιδί και αν είναι το αντικαθιστώ. Γράφω στο αρχείο του γονιού αν πρόκειται για αντικατάσταση του segment ή φόρτωση (αν φορτώνουμε πρώτη φορά segment) μαζί με το χρόνο αντικατάστασης ή φόρτωσης. Για την αντικατάσταση όλου του segment θα χρειαστεί να διαβάσω χαρακτήρες μέχρι και τον τελευταίο χαρακτήρα του προηγούμενου segment. Τέλος για κάθε γραμμή του segment παίρνω τους χαρακτήρες από το αρχείο(αν είμαστε στο τελευταίο segment ελέγχουμε αν έχουμε πιο λίγες γραμμές). Ο

γονιός μετά στέλνει απάντηση στο παιδί για το request που μόλις διάβασε και αντίγραψε με χρήση σημαφόρων response_parent,response_child.

8. Ο γονιός περιμένει μέχρι να τελειώσουν όλες οι διεργασίες παιδιά για να καταστρέψει και να ελευθερώσει τους σημαφόρους, τα αρχεία και τα shared memory.

Στην childProccess:

1. Το παιδί ανοίγει το δικό του αρχείο (1 για κάθε παιδί) για να γράψει τα αποτελέσματα.
2. Αποδεσμεύει κάθε φορά τα shared responses που δεν χρειάζεται και ταυτίζω το response για το συγκεκριμένο παιδί με την αντίστοιχη θέση του shared response(ένας πίνακας με όλα τα struct responses για κάθε παιδί).
3. Έχουμε ένα for loop με τόσες επαναλήψεις όσα και τα αιτήματα του κάθε παιδιού. Μέσα στη κάθε επανάληψη δημιουργούμε τυχαία το segment (με πιθανότητα να έχουμε το ίδιο segment 0.7) και τυχαία την γραμμή(αν είμαστε στο τελευταίο segment ελέγχουμε πόσες γραμμές έχει). Έτσι δημιουργείται το κάθε request του παιδιού. Στη συνέχεια έχουμε το πρωτόκολλο producer-consumer αντίστοιχα για τη διεργασία παιδί τώρα όπου βάζουμε το request που μόλις φτιάξαμε στην αντίστοιχη θέση του buffer(in) και μεταφέρουμε τον δείκτη στην επόμενη θέση. Ταυτόχρονα κρατάμε χρόνο δημιουργίας του request. Πιο κάτω το παιδί παραλαμβάνει το request από τον γονιό και γράφει στο αρχείο του το χρόνο που έστειλε το request, το χρόνο response,το ίδιο το request και τη γραμμή που αντιστοιχεί. Τέλος η διεργασία παιδί αναμένει ένα χρόνο 20 ms για να τερματίσει .
4. Αφού ολοκληρωθούν όλα τα requests του παιδιού, κάνουμε αποσύνδεση από το response, κλείνουμε το αρχείο και το παιδί τερματίζει.

Εντολές στο terminal: 4 παιδιά,5 γραμμές ανά παιδί, 5 request το κάθε παιδί

gcc -o ergasia1 ergasia1.c -lpthread

./ergasia1 text.txt 4 5 5