ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΕΡΓΑΣΙΑ 1

Κώτσιας Γιάννης
1115202000113

Η εργασία αποτελείται συνολικά από 7 modules για την καλύτερη οργάνωση του κώδικα, τα οποία παρουσιάζονται παρακάτω.

1. parent.cpp
2. child.h
3. child.cpp
4. utils.h
5. utils.cpp
6. Makefile
7. text.txt

Η εργασία είναι υλοποιημένη με τη γλώσσα C++. Δεν έχει χρησιμοποιηθεί η βιβλιοθήκη STL, καθώς γίνεται κυρίως χρήση βιβλιοθηκών της γλώσσας C.

Εκτέλεση

1. Στο terminal ανοίξτε με τα κατάλληλα cd τον τρέχων κατάλογο της εργασίας και εκτελέστε τις παρακάτω εντολές
2. make
3. ./parent <file> <N> <Βαθμός_κατάτμησης > <Πλήθος_Αιτήσεων_ανά_Παιδί >

Ενδεικτική εκτέλεση: ./parent text.txt 10 10 5

Output

1. Για κάθε διεργασία παιδί παράγεται ένα αρχείο καταγραφής με όνομα child_N, όπου N ο αριθμός αναγνωριστικού του παιδιού.
2. Η μητρική διεργασία παράγει ένα αρχείο καταγραφής με όνομα parent_file.txt.
3. Το κάθε αρχείο καταγραφής παιδιού περιέχει το χρόνο υποβολής του αιτήματος, το χρόνο απάντησης, τον αριθμό ζητούμενου segment, τον αριθμό ζητούμενης γραμμής και την ζητούμενη γραμμή από το αρχείο.
4. Το αρχείο καταγραφής της μητρικής εργασίας περιέχει τον αριθμό του segment, τον αριθμό γραμμής, το χρόνο υποβολής του αιτήματος και το χρόνο απάντησης.

Υλοποίηση

utils

Στο συγκεκριμένο αρχείο υλοποιούνται:

1. *struct* ***shared memory***. Σε αυτή περιέχονται:
    a. Ο πίνακας σεμαφόρων *semPerSegMutex* μεγέθους (sem) * *numberOfSegments* ο οποίος περιέχει ένα σεμαφόρο για κάθε segment
    b. Ο σεμαφόρος *FIFOmutex* του οποίου η χρήση είναι οι διεργασίες παιδιά να εξυπηρετούνται με σειρά FIFO
    c. Ο σεμαφόρος *request*
    d. Ο σεμαφόρος wait_*response*
    e. Ο σεμαφόρος *write*
    f. Ο πίνακας ακεραίων *countOfReaders* μεγέθους (int)**numberOfSegments* ο οποίος περιέχει τον αριθμό των readers για κάθε segment
    g. Ο ακέραιος *currentReaderCounter*, ο οποίος περιέχει συνολικό αριθμό των readers
    h. Ο ακέραιος *segmentRequested*, ο οποίος υποδεικνύει τον ζητούμενο αριθμό του segment
    i. Ο ακέραιος *lineRequested*, ο οποίος υποδεικνύει τον ζητούμενο αριθμό της γραμμής
    j. Ο ακέραιος *currentSegment*, ο οποίος υποδεικνύει τον τρέχων αριθμό του segment στη διαμοιραζόμενη μνήμη
2. *char*** ***start****(char* argv, int linesPerSegment, int* numberOfLines, int* lengthOfLine)*
    a. Επιστρέφεται στον γονέα ο 3x3 πίνακας char*** arrayOfSegments[segment][linesOfSegment][line] ο οποίος χωρίζει το αρχείο εισόδου σε τμήματα για πιο καθαρή και ξεκάθαρη υλοποίηση
    b. Αφού δημιουργηθεί ο πίνακας και αντιγραφεί το αρχείο εισόδου σε αυτόν τότε αυτό διαγράφεται για να μην επιβαρύνεται η μνήμη με περιττά δεδομένα
    c. Επιστρέφει στον γονέα τη μεταβλητή numberOfLines η οποία ορίζει το μέγιστο αριθμό γραμμών στο αρχείο εισόδου
    d. Επιστρέφει στον γονέα τη μεταβλητή lengthOfLine η οποία ορίζει το μέγιστο αριθμό χαρακτήρων ανά γραμμή στο αρχείο εισόδου

3. *int **random_segment**(int min, int max)*
    a. Επιστρέφει ακέραιο αριθμό στο εύρος [min, max]
4. *int **random_line**(int min, int max)*
    a. Επιστρέφει ακέραιο αριθμό στο εύρος [min,max)

child

void child(int segmentID, int numberOfRequests, int linesPerSegment, int numberOfSegments, int lengthOfLine, SharedMemory sharedMem, char buffer)*

1. Υλοποιεί τη διεργασία του παιδιού
2. Ως ορίσματα δέχεται
    a. Αναγνωριστικό αριθμό του δοθέντως segment
    b. Αριθμό request για το κάθε παιδί
    c. Αριθμό γραμμών του κάθε segment
    d. Συνολικό αριθμό των segments
    e. Συνολικό αριθμό χαρακτήρων ανά γραμμή
    f. Δείκτη στην διαμοιραζόμενη μνήμη
    g. Δείκτη στο πίνακα κοινής μνήμης που μεταφέρει την αιτούμενη γραμμή από μητρική διεργασία στο παιδί
3. Δημιουργεί ένα νέο αρχείο καταγραφής με όνομα child_N με N στο εύρος [0, N) το οποίο θα εκμεταλευτεί στο τέλος της διεργασίας το παιδί
4. Επιλέγει τυχαία αιτούμενο αριθμό γραμμής του segment
5. Επιλέγει τυχαία αιτούμενο αριθμό segment με πιθανότητα 70% ίδιο με τον προηγούμενο και 30% διαφορετικό
6. Υλοποιεί το μοντέλο συγχρονισμού Readers – Writers με σεμαφόρους *semPerSegMutex[targetSegment], write, request, wait_response*
7. Αντιγράφει στον δικό του target_buffer το περιεχόμενο της αιτούμενης γραμμής από τον buffer της διαμοιραζόμενης μνήμης
8. Υπολογίζει τον χρόνο υποβολής του αιτήματος και τον χρόνο απάντησης
9. Τυπώνει στο αρχείο καταγραφής τους παραπάνω χρόνους, τους αριθμούς segment και line στόχου και τέλος την αιτούμενη γραμμή

parent

1. Δημιουργεί τη shared memory
2. Αρχικοποιεί τα περιεχόμενα της
    a. Σεμαφόροι
    b. Πίνακες
    c. Ακέραιοι
3. Δημιουργεί τον διαμοιραζόμενο στην κοινή μνήμη buffer που μεταφέρει τα αιτήματα από την μητρική διεργασία προς τα παιδιά
4. Εκκινεί Ν διεργασίες παιδιά
5. Δημιουργεί αρχείο καταγραφής *parent_file.txt*
6. Τυπώνει σε αυτό τους τελευταίους αριθμούς segment και γραμμής που αιτήθηκαν καθώς και τους χρόνους εξυπηρέτησης των θυγατρικών διεργασιών
7. Αντιγράφει στον shared buffer το αιτούμενο segment sharedMem -> segmentRequested και γραμμή sharedMem -> lineRequested
8. Δίνει τη σειρά στη διεργασία παιδί
9. Περιμένει να τελειώσουν όλες οι διεργασίες παιδιά
10. Καταστρέφει την κοινή μνήμη και τα περιεχόμενά της
11. Καταστρέφει τον buffer κοινής μνήμης
12. Καταστρέφει τον πίνακα από segments του αρχείου εισόδου
13. Τέλος εργασίας.