ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ – ΕΡΓΑΣΙΑ 1

ΚΩΝΣΤΑΝΤΙΝΟΣ ΣΤΑΓΚΟΣ 1115201600159

Το πρόγραμμα το οποίο δημιούργησα αρχικά δέχεται τέσσερα ορίσματα με την αντίστοιχη σειρά που αναφέρεται στην εκφώνηση και τα αποθηκεύει στις αντίστοιχες μεταβλητές. Στη συνέχεια ανοίγω το αρχείο κειμένου από τη γονική διεργασία και μετράω τις γραμμές του αρχείου όπου τις εμφανίζω με αντίστοιχη εκτύπωση. Έπειτα κάνω rewind το αρχείο ώστε την επόμενη φορά να το διατρέξω πάλι από την αρχή. Τέλος υπολογίζω από πόσες γραμμές θα αποτελείται κάθε segment.

Εφόσον πλέον έχω διαχειριστεί κατάλληλα το αρχείο δημιουργώ διαμοιραζομένη μνήμη η οποία ως δομή (struct) αποτελείται από ένα πίνακα χαρακτήρων εκατό θέσεων για να περνάω τη κάθε γραμμή του αρχείου καθώς και δύο μεταβλητές ακέραιων για να γνωστοποιώ στις διεργασίες παιδιά τις γραμμές του αρχείου καθώς και το ποιο segment βρίσκεται τη συγκεκριμένη στιγμή μέσα στη μνήμη. Ταυτόχρονα δημιουργώ και τρεις απαιτούμενους σημαφόρους για τον συγχρονισμό των διεργασιών χρησιμοποιώντας την συνάρτηση createSem η οποία δέχεται ως όρισμα τη τιμή αρχικοποίησης του σημαφόρου καθώς και το id της shared memory.Οι σημαφόροι μας βοηθούν στη διαχείριση και στον συγχρονισμό των διεργασιών καθώς και ελέγχουν την είσοδοέξοδο στη shared memory.Ο σημαφόρος childSemId (αρχικοποιείται στο 0) είναι) είναι υπεύθυνος να διακόπτει την διεργασία παιδί όταν χρειάζεται αντίστοιχα ο fatherSemId (αρχικοποιείται στο 0) είναι) είναι υπεύθυνος για την γονική διεργασία ενώ ο sharedMemorySemId καθορίζει το ποιος θα μπαίνει και θα γράφει μέσα στη διαμοιραζομένη μνήμη.

Στη συνέχεια δημιουργώ με μια for τα αντίστοιχα παιδιά που μου ζητήθηκαν ως όρισμα. Όταν είμαι στη διεργασία παιδί κατεβάζω τον σημαφόρο sharedMemorySemId τον οποίο έχω αρχικοποιήσει με μονάδα ώστε να μπορεί να μπει το παιδί που θα προλάβει να τον κατεβάσει και να κλειδώσει τη shared memory για τα επόμενα ώστε να μπορεί να γράψει και να πάρει πληροφορίες μόνο αυτό. Έπειτα με τη βοήθεια της γεννήτριας rand ,στην οποία δίνω ως όρισμα το στην οποία δίνω ως όρισμα το id του κάθε παιδιού για να παράγει διαφορετικούς και τυχαίους αριθμούς , παίρνω ένα τυχαίο ακέραιο από το 1 έως το πλήθος των γραμμών του αρχείου ενός segment και έναν τυχαίο ακέραιο από το 1 έως το πλήθος των segments του κειμένου. Στη συνέχεια ξέροντας το segment και τη γραμμή που χρειαζόμαστε, υπολογίζουμε το ποια γραμμή είναι μέσα στο κείμενο και καταχωρώ τη τιμή αυτή στην shared memory έπειτα ενεργοποιώ τον πατερά ανεβάζοντας τον σημαφόρο fatherSemId και μπλοκάρω το παιδί κατεβάζοντας τον σημαφόρο childSemId. Αυτό συμβαίνει για να διακόψει η διεργασία παιδί και να

τρέξει η γονική διεργασία να δει τη γραμμή που ζητάει το αντίστοιχο παιδί να την καταχωρήσει στην shared memory και να ανεβάζει το σημαφόρο childSemId ώστε να ενεργοποιηθεί το παιδί και να κατεβάσει τον αντίστοιχο ώστε να μπλοκαριστεί η γονική. Έτσι το παιδί το οποίο βρίσκεται αυτή τη στιγμή στη κρίσιμη περιοχή μπορεί να συνεχίσει και να βρει την γραμμή την οποία ζήτησε στην αντίστοιχη μεταβλητή και να την εκτυπώσει. Εφόσον έχει τελειώσει τα αιτήματα που επιθυμούσε εκτυπώνουμε του χρόνους εισαγωγής κάθε αιτήματος καθώς και τον χρόνο εξυπηρέτησης τους. Τέλος ξεκλειδώνει τον πατερά εφόσον έχει ολοκληρώσει και ο πατέρας με τη σειρά του ανεβάζει τον σημαφόρο sharedMemorySemId ώστε να συνεχίσει η διαδικασία με το επόμενο παιδί.

Η παραπάνω διαδικασία γίνεται για K παιδιά και για N φορές για το κάθε παιδί όπως ακριβώς μας ζητείται. Έχω βάλει τη γονική διεργασία σε μια for η οποία κάνει N*K επαναλήψεις αυτό το επέλεξα διότι ο γονιός συμπεριφέρεται ως εξυπηρετητής οπότε θα αναγκαστεί να τρέξει για κάθε παιδί και για κάθε δοσοληψία. Επίσης η τεχνική αυτή με εξυπηρετεί στο να μην γεννιέται ένα παιδί και να εξυπηρετείται για N φορές και έπειτα να δημιουργείται το δεύτερο. Με τον τρόπο αυτό καταφέρνω τα παιδιά να τρέχουν παράλληλα και να εξυπηρετείται αυτό που θα προλάβει να κάνει down τον σημαφόρο sharedMemorySemId. Τέλος περιμένω όλα τα παιδιά να ολοκληρώσουν τη διαδικασία και απελευθερώνω την μνήμη των σημαφόρων και της διαμοιραζομένης.

*Στην αρχή του child process έχω δημιουργήσει έναν πίνακα από int όπου εκεί αποθηκεύονται τα κομμάτια(segments) που αναφέρεται κάθε αίτημα, ενώ στο τέλος εκτυπώνω τον πίνακα αυτόν. Αυτό δημιουργήθηκε για να με εξυπηρετήσει στο κομμάτι παράλληλης εξυπηρέτησης πολλών παιδιών σε πιθανό αίτημα κοινού τμήματος κειμένου, υλοποίηση που δεν ολοκληρώθηκε εν τέλει.

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ - ΕΡΓΑΣΙΑ

Σύντομες τεχνικές λεπτομέρειες:

• Το αρχείο που απέστειλα αποτελείται από ένα αυτοδημιούργητο αρχείο κειμένου δυο αρχεία .c(os.c,στην οποία δίνω ως όρισμα το os_funcs.c και ένα αρχείο .h(os_funcs.h) και ένα Makefile.

• Στο os.c βρίσκεται η main συνάρτηση του προγράμματος που υλοποίησα με αντίστοιχα σχόλια επεξήγησης σχεδόν σε κάθε γραμμή.

• Στο os_funcs.c βρίσκονται οι συναρτήσεις τις οποίες χρησιμοποίησα στη main

- Στο os_funcs.h βρίσκονται οι δομές που χρησιμοποίησα για υλοποίηση καθώς και οι επικεφαλίδες των συναρτήσεων

Συμπληρωμάτικες λεπτομέρειες:

- Για την υλοποίηση της ασκήσεις έχω χρησιμοποιήσει συναρτήσεις για την δημιουργία, στην οποία δίνω ως όρισμα το αρχικοποίηση, στην οποία δίνω ως όρισμα το ανέβασμα και κατέβασμα καθώς και διαγραφή σημαφόρων από το κώδικα φροντιστηρίου που μας δόθηκε.

- Επίσης για την μέτρηση του χρόνου εκτέλεσης κάθε διεργασίας από την υποβολή ενός αιτήματος μέχρι τη λήψη της αντίστοιχης απάντησης με τη χρήση μεταβλητών τύπου clock_t

- Στην προσπάθεια μου να διατρέξω το αρχείο κειμένου και να πάρω τη γραμμή που επιθυμώ χρησιμοποίησα τη συνάρτηση getline().

Ενδεικτικός τρόπος εκτέλεσης:

- make

- ./os file 100 15 100(όπου 100 είναι τα παιδιά, 15 είναι οι αιτήσεις και 100 είναι το σε πόσα κομμάτια (segments) θα σπάσουμε το αρχείο

- Μπορεί να χρησιμοποιηθεί και η εντολή make clean για διαγραφή των εκτελέσιμων