ΧΑΡΙΣΗΣ ΝΙΚΟΛΑΟΣ 1115201700187

Η εργασία αποτελείται από τα αρχεία misc.h misc.c server_program.c client_program.c input.txt και ένα Makefile. Για καθένα από αυτά θα δοθούν παρακάτω περαιτέρω σχόλια και σημειώσεις.

Χρησιμοποιούνται 3 σημαφόροι, καθώς και το μοντέλο server-client για τα ζητούμενα τις εργασίας. Η διαμοιραζόμενη/κοινή μνήμη είναι ένα struct που περιέχει 1 ακέραιο(εκεί γράφουν τα παιδιά το αίτημά τους και από εκεί διαβάζει ο πατέρας το αίτημα ενός παιδιού) και 1 συμβολοσειρά(εκεί γράφει ο πατέρας την απάντησή του στο αίτημα του παιδιού και από εκεί διαβάζει το παιδί την απάντηση του πατέρα στο αίτημά του).Ο $1^{ος}$ (server_sem)από τους 3 σημαφόρους χρησιμοποιείται για την επικοινωνία του πατέρα με την διαμοιραζόμενη μνήμη(πότε μπορεί να γράψει/διαβάσει ο πατέρας), ο $2^{ος}$ (client_sem)για την επικοινωνία ενός παιδιού με την διαμοιραζόμενη μνήμη(πότε μπορεί ένα παιδί να γράψει/διαβάσει) και ο $3^{ος}$ (inter_client_sem) καθορίζει ποιο από τα παιδιά που δεν έχουν κάνει exit μπορεί να επικοινωνήσει με τη διαμοιραζόμενη μνήμη(ποιο παιδί μπορεί να γράψει/διαβάσει).Γενικά για την χρήση ,την αρχικοποίηση και την καταστροφή των σημαφόρων χρησιμοποιείται το Sys V ενώ για την αρχικοποίηση, την

χρήση και τη καταστροφή της διαμοιραζόμενης μνήμης χρησιμοποιείται το IPC.

misc.h:

Περιέχει διάφορα includes & enums που απαιτούνται για την εύρυθμη χρήση του Sys V και του IPC. Ορίζει τη δομή της διαμοιραζόμενης μνήμης(struct memory) καθώς και το μέγιστο μέγεθος της γραμμής του αρχείου X(input.txt στα πλαίσια των δικών μου εκτελέσεων) με το LINE_SIZE(101 αφού θέλουμε 100 bytes για τους 100 χαρακτήρες της γραμμής και 1 ακόμη για το \0). Τέλος, περιέχει τα πρότυπα ποικίλων συναρτήσεων που χρησιμοποιούνται τόσο από το server_program.c όσο και από το client_program.c.

Οι συναρτήσεις sem_P,sem_V,sem_Init,free_resources είναι από τον κώδικα του φροντιστηρίου που έχει δοθεί στο eclass(με μικρές αλλαγές στις εκτυπώσεις σφαλμάτων).

misc.c:

Περιέχει την υλοποίηση των συναρτήσεων που περιγράφονται στο misc.h. Για τις συναρτήσεις που είναι όμοιες με των κώδικα του φροντιστηρίου, δεν υπάρχουν ιδιαίτερα σχόλια πέρα ίσως από το γεγονός ότι τα μηνύματα σε περίπτωση σφάλματος έχουν αλλάξει σε κάποιες περιπτώσεις για να ταιριάζουν με το ύφος των εκτυπώσεων στο server_program.c. Αν

υπάρχει ένα άλλο στοιχείο που είναι άξιο αναφοράς, είναι το γεγονός ότι η get_file_lines δεν ανοίγει κάποιο αρχείο, για αυτό προκείμενου να δουλέψει θα πρέπει το αρχείο να έχει ανοιχτεί από πριν. Στην δικιά μας περίπτωση δεν συναντάμε προβλήματα, απλά αν θέλαμε η συνάρτηση αυτή να είναι πιο γενικού σκοπού θα έπρεπε να δέχεται σαν όρισμα το όνομα του αρχείου και να το ανοίγει η ίδια. Επίσης, σε περίπτωση που δεν έχει ανοιχτεί το αρχείο επιστρέφει μηδέν.

server_program.c:

Εδώ υλοποιείται ο γονέας σαν server. Στην αρχή υπάρχουν διάφοροι έλεγχοι για να βεβαιωθούμε ότι η είσοδος είναι η αναμενόμενη. Αν δεν είναι, τερματίζουμε αφού το υπόλοιπο πρόγραμμα εξαρτάται από τη σωστή είσοδο. Επίσης, ανοίγουμε το αρχείο, βρίσκουμε πόσες γραμμές έχει και κάνουμε τον αντίστοιχο έλεγχο(μετά τον έλεγχο αυτό ξανανοίγουμε το αρχείο αφού για να βρούμε τις γραμμές του έπρεπε να φτάσουμε στο τέλος του). Στην συνέχεια δημιουργούμε ένα κλειδί μέσω της ftok ,το οποίο θα χρησιμοποιήσουμε για να φτιάξουμε την διαμοιραζόμενη μνήμη και τους 3 σημαφόρους(πιο σωστά για να φτιάξουμε 3 διαφορετικούς σημαφόρους θα χρησιμοποιήσουμε το key+1,key+2,key+3 αντίστοιχα ,το οποίο μας βοηθάει να καταλάβουμε σε ποιο σημαφόρο αναφερόμαστε). Έπειτα, δημιουργούμε την

διαμοιραζόμενη μνήμη και την κάνουμε attach.Αν κάποια από της παραπάνω δραστηριότητες αποτύχει γυρνάμε το αντίστοιχο μήνυμα λάθους και τερματίζουμε. Μέτα προχωράμε στην δημιουργία και αρχικοποίηση των τριών σημαφόρων ξεκινώντας από τον server_sem, μετά τον client_sem και τέλος τον inter_client_sem. Αν σε κάποια φάση αποτύχουμε κάνοντας τα προηγούμενα, αποδεσμεύουμε την μνήμη (αφού την κάνουμε detach) και καταστρέφουμε όσους σημαφόρους έχουμε δημιουργήσει μέχρι τότε. Αυτή είναι μία διαδικασία που είναι αποκλειστική στον server/πατέρα. Έπειτα, προετοιμαζόμαστε να δημιουργήσουμε Κ παιδία και να γνωστοποιήσουμε σε αυτά τα απαραίτητα στοιχεία που πήραμε από την είσοδο. Έτσι δεσμεύουμε 5 string(s_i), στα οποία μέσω της sprintf περνάμε τις απαραίτητες πληροφορίες. Τα string αυτά τα εισάγουμε στο string array arg το οποίο θα χρησιμοποιηθεί για την κλήση της execv κατά τη δημιουργία των Κ παιδιών(τον αριθμό των δοσοληψιών δεν χρειάζεται να τον πειράξουμε μέσω sprint αφού το έχουμε από την είσοδο σαν argv[3]). Όπως φαίνεται και στα σχόλια του κώδικα κάθε θέση του πίνακα arg εκφράζει συγκεκριμένη πληροφορία. Ειδική μνεία χρειάζεται στη θέση 0 που έχει το όνομα του εκτελέσιμου, καθώς και στη θέση 7 που έχει NULL γιατί έτσι απαιτεί το execv. Έπειτα από αυτό , ο πατέρας προχωρά στην δημιουργία των Κ παιδίων. Αν το pid

είναι 0 , μεταφερόμαστε μέσω της execv στο κώδικα των παιδιών. Αν είναι αρνητικό , επιστρέφουμε μήνυμα λάθους και τερματίζουμε.

Για τον πατέρα η μεταβλητή demand, που δείχνει στο get_line της κοινής μνήμης, αποτελεί το αίτημα του παιδιού ενώ η μεταβλητή answer,που δείχνει στο line της κοινής μνήμης, αποτελεί την απάντησή του στο αίτημα του παιδιού. Έτσι, αφού πρώτα υπολογίσουμε τις συνολικές δοσοληψίες στις οποίες θα λάβει μέρος ο πατέρας, πάμε να τις ικανοποιήσουμε μια μια. Αρχικά , κάνουμε down τον server_sem(sem_P(server_sem)) για να μπορέσουμε να διαβάσουμε ποια γραμμή θέλουμε (το αίτημα). Αφού διαβάσουμε, εντοπίζουμε αυτή τη γραμμή μέσω της εμφωλευμένης επανάληψης που υπάρχει και την αντιγράφουμε στο answer και συνεπώς και στο line της κοινής μνήμης. Αφού τελειώσαμε, επιτρέπουμε στα παιδιά να αλληλεπιδράσουν με τη μνήμη κάνοντας up τον client_sem (sem_V(client_sem)). Αφού ικανοποιήσουμε όλα τα αιτήματα παριμένουμε τα παιδιά να τερματίσουν. Αφού συμβεί και αυτό, κάνουμε detach και καταστρέφουμε την κοινή μνήμη και τους 3 σημαφόρους. Τέλος, τερματίζουμε.

client_program.c:

Πρώτα, αρχικοποιούμε την srand με seed το id κάθε παιδιού, το οποίο είναι διαφορετικό για όλα τα παιδιά(η time() δεν είναι ιδιαίτερα χρήσιμη αφού ο

χρόνος που περνάει από κάθε κλήση της είναι πολύ μικρός με συνέπεια να μην έχουμε καλή τυχαιότητα). Έπειτα, μετατρέπουμε την είσοδο από string σε int μέσω της atoi(). Στις μεταβλητές demand(που δείχνει στο get_line της κοινής μνήμης) και answer(που δείχνει στο line της κοινής μνήμης) φυλάσσονται αντίστοιχα το αίτημα του παιδιού και η απάντηση του πατέρα. Η μεταβλητή get_line που ακολουθεί , χρησιμοποιείται για ν επιλέξουμε τυχαία μια γραμμή του αρχείου Χ. Να σημειωθεί ότι προς το παρόν δεν έχουμε κανένα έλεγχο για errors,με εξαίρεση κατά τα attach της κοινής μνήμης ,αφού η client_program καλείται αποκλειστικά από το server_program με σωστό τρόπο. Επίσης , χρησιμοποιούμε της μεταβλητές el_time,begin,end τύπου clock_t για τον υπολογισμό του χρόνου που απαιτείται για την εκτέλεση των Ν δοσοληψιών και ,κατά συνέπεια, για το μέσο χρόνο εκτέλεσης μιας δοσοληψιας του παιδιου.

Κάθε παιδί εκτελεί Ν δοσοληψίες. Έτσι, για κάθε δοσοληψία αρχικά προλάβαινουμε τα άλλα παιδιά πριν αλληλεπιδράσουν με την μνήμη κάνοντας down τον inter_client_sem(sem_P(inter_client_sem)).Έπειτα , καταθέτουμε το αίτημά μας κάνοντας down τον client_sem (sem_P(client_sem)). Μετά, διαλέγουμε ποια γραμμή θα αιτηθούμε ενώ «παράλληλα» αρχίζουμε την χρονομέτρηση. Γράφουμε το αίτημά μας στην κοινή μνήμη και κανουμε up τον sever_sem

προκειμένου να μπορέσει να απαντήσει ο server(sem_V(server_sem)). Ζητάμε να διαβάσουμε την απάντηση στ αίτημά μας κάνοντας down τον client_sem(sem_P(client_sem)). Αφού το διαβάσαμε την απάντηση στο αίτημά μας, σταματάμε την χρονομέτρηση. Ο συνολικό χρόνος για να ικανοποιηθεί το αίτημά μας είναι dt=end-begin.(Να σημειωθεί ότι πρακτικά δεν χρονομετρούμε αλλά μετρούμε σε ποιο κύκλο αρχίσαμε και σε ποιο κύκλο τελειώσαμε, έτσι ώστε αν τους αφαιρέσουμε να βρούμε πόσοι κύκλοι μεσολάβησαν από την κατάθεση του αιτήματος μέχρι την απάντησή του). Προσθέτουμε στο συνολικό χρόνο το χρόνο που μόλις βρήκαμε ότι μας πήρε αυτή η διεργασία(dt) και συνεχίζουμε κάνοντας up τον inter_client_sem έτσι ώστε να αρχίσουν να συναγωνίζονται τα υπόλοιπα παιδιά που έχουν στόχο να αλληλεπιδράσουν με την κοινή μνήμη(sem_V(inter_client_sem)).Τέλος, αφού η δοσοληψία τελείωσε κάνουμε up τον client_sem(sem_V(client_sem)) , έτσι ώστε το τυχερό παιδί που θα πάρει το OK από το inter_client_sem να μπορέσει να αλληλεπιδράσει με την μνήμη.

Αφού ολοκληρωθούν όλες οι δοσοληψίες , βρίσκουμε πόσο χρόνο τους πήρε διαιρώντας το πλήθος των κύκλων el_time με το CLOCKS_PER_SEC. Προφανώς για το μέσο χρόνο διαιρούμε την ποσότητα που βρήκαμε(total_time) με το πλήθος των

δοσοληψιών(n_tactions). Τέλος, εκτυπώνουμε τα ζητούμενα και τερματίζουμε.

Makefile:

Το Makefile περιέχει πέρα από τις recipes για τα διάφορα εκτελέσιμα και ένα clean το οποίο χρησιμοποιήθηκε κατά το debugging.

Τρέχοντας make project1 δημιουργούνται τα server_program client_program. Μια τυπική εκτέλεση που προτείνω είναι η < ./server_program input.txt 25 100 > , η οποία διαβάζει από το αρχείο input.txt που παραδίδεται και δημιουργεί 25 παιδιά ,καθένα εκ των οποίων συμμετέχει σε 100 δοσοληψίες.

input.txt:

23 από τις καλύτερες ατάκες που ακούγονται κατά τη διάρκεια της prequel τριλογίας του Star Wars(οι περισσότερες είναι από το Episode 3 και αναφέρονται με τον έναν ή τον άλλο τρόπο στον Anakin).