Λειτουργικά Συστήματα 1η Εργασία 2021-2022

Αλατζάς Αλέξανδρος

Α.Μ.: 1115201900005

Εντολή μεταγλώττισης και εκτέλεσης: make run

Περιεχόμενα: OS_ergasia1.c , Makefile, README_ergasia1.pdf, Stairway.txt, (εκφώνηση)

-Stairway.txt: Ένα ενδεικτικό αρχείο κειμένου που χρησιμοποιήθηκε ως το αρχείο (X) για τις δοκιμές της εργασίας. Περιέχει τους στίχους του τραγουδιού Stairway to Heaven καθώς και μια έξτρα σειρά στο τέλος του αρχείου για να φανεί ότι εκτυπώνονται το πολύ 100 χαρακτήρες. Συγκεκριμένα, αν τύχει η εν λόγω σειρά παρατηρείστε ότι δεν εκτυπώνεται ο χαρακτήρας της τελείας.

-README_ergasia1.pdf: Το παρών κείμενο

-Makefile: Ένα απλό Makefile για την μεταγλώττιση και την εκτέλεση του προγράμματος. Εκεί μέσα καλείται όποιος επιθυμεί να εκτελέσει το πρόγραμμα, να αλλάξει κατά βούληση τα ορίσματα (X), (K), (N). Ενδεικτικά είναι ορισμένα στο Stairway.txt, 10, 10. Δηλαδή για 10 διεργασίες-παιδιά και 10 δοσοληψίες για το καθένα.

-OS_ergasia1.c: Το αρχείο πηγαίου κώδικα. Όλη η εργασία εκπονήθηκε σε ένα .c αρχείο. Ακολουθεί η επεξήγησή της. (Συμπληρώνουν και σχόλια στον κώδικα)

Τεχνικές λεπτομέρειες: Αρχικά, ορίζω με #define το μέγεθος της γραμμής κειμένου που θα χρησιμοποιώ σε 102(bytes) έτσι ώστε να πάρω το πολύ 100 χαρακτήρες από το κείμενο. Στις δοσοληψίες, ορίζω το 101^{ο} και 102^{ο} χαρακτήρα της γραμμής στους αντίστοιχους αλλαγής γραμμής '\n' και τέλους string '\0'. Έπειτα ορίζω κάποια keys για τους σημαφόρους και την διαμοιραζόμενη μνήμη. Φτιάχνω μια struct που θα αναπαριστά την διαμοιραζόμενη μνήμη, η οποία αποτελείται από έναν πίνακα χαρακτήρων για την γραμμή του κειμένου καθώς και έναν ακέραιο για τον αριθμό της γραμμής. Ορίζω και μία συνάρτηση η οποία επιστρέφει την τιμή ενός σημαφόρου.

Στο κυρίως πρόγραμμα, ανοίγω το αρχείο σύμφωνα με τα ζητούμενα τις εκφώνησης, ώστε η γονική διεργασία να καταμετρήσει τον αριθμό των γραμμών του αρχείου. Κατόπιν, δημιουργώ την διαμοιραζόμενη μνήμη, την κάνω attach και δημιουργώ 4 named σημαφόρους. Οι σημαφόροι ταυτίζονται με τις εξής λειτουργίες: ο sem1 χρησιμοποιείται για τον έλεγχο της διεργασίας-παιδί, ο sem2 αντίστοιχα για την μητρική, ο sem3 εξασφαλίζει τον αμοιβαίο αποκλεισμό, ότι δηλαδή δεν θα επιχειρεί κάποια διεργασία να αποκτήσει πρόσβαση στη

διαμοιραζόμενη μνήμη όσο εκτελεί την λειτουργία της κάποια άλλη, ο sem4 έχει στην ουσία τον ρόλο του μετρητή, θα δώσει το σήμα αφού όλες οι διεργασίες παιδιά ολοκληρώσουν, ώστε να κάνει wait() η μητρική.

Η μητρική διεργασία δημιουργεί τα Κ παιδιά. Αυτά μετά τη δημιουργία τους, διαμορφώνουν τα αιτήματα προς τη μητρική. Η κάθε διεργασία παιδί έχει ένα for loop για τις Ν δοσοληψίες που έχουμε ορίσει, ωστόσο κλειδώνει και περιμένει τον sem3 να γίνει post(). Ο sem3 αρχικοποιήθηκε με τιμή 1, άρα την πρώτη φορά δεν περιμένει, κατεβάζει την τιμή του και συνεχίζει την εκτέλεση. Εκεί θα ξεκινήσει την χρονομέτρηση του αιτήματος, θα ορίσει τον τυχαίο αριθμό γραμμής, θα τον γράψει στην διαμοιραζόμενη μνήμη και θα εκτυπώσει αναγνωριστικό μήνυμα. Σηκώνει τον σημαφόρο sem2 και κλειδώνει τον sem1, οπότε περιμένει τον sem1 να σηκωθεί. Η προσοχή μας τώρα στρέφεται στον κώδικα της μητρικής διεργασίας. Εκεί η μητρική περίμενε διαρκώς μέχρι να λάβει το πρώτο της αίτημα. Το ανέβασμα το sem2 σηματοδότησε το αίτημα, οπότε συνέχισε στην εκτέλεσή της. Εκεί καταχωρεί σε μία τοπική μεταβλητή τον ζητούμενο αριθμό γραμμής, ανοίγει ξανά το αρχείο (Χ) και το διαβάζει γραμμή προς γραμμή μέχρι να βρει την ζητούμενη γραμμή. Τότε, κάνει την τροποποίηση που προαναφέραμε ώστε να έχουμε το πολύ 100 χαρακτήρες κειμένου και περνάει τους χαρακτήρες από τον τοπικό της πίνακα στην διαμοιραζόμενη μνήμη. Εκτυπώνει το αναγνωριστικό της μήνυμα και σηκώνει τον σημαφόρο sem1. Γυρνάμε πάλι στον κώδικα της διεργασίας-παιδί. Τώρα που ο sem1 σηκώθηκε, τον κατεβάζει πάλι και διαβάζει από τη διαμοιραζόμενη μνήμη όσα έδωσε η μητρική διεργασία. Αφού τα περάσει σε τοπικές μεταβλητές, δεν χάνει χρόνο και σηκώνει τον σημαφόρο sem3 ώστε να λάβει τον έλεγχο η επόμενη διεργασία, όσο η ίδια τυπώνει το αναγνωριστικό της μήνυμα. Αξίζει να σημειωθεί πως αρκετές φορές μπορεί να τύχει να προλάβει ξανά η ίδια διεργασία τον έλεγχο καθώς αυτό εξαρτάται και από το σύστημα.

Όταν μία διεργασία-παιδί τελειώσει με όλες τις δοσοληψίες της, υπολογίζει τον μέσο χρόνο που χρειάστηκε για την ικανοποίηση των αιτημάτων. Ο χρόνος μετρείται σε clocks ανά δευτερόλεπτο και τυπώνω τον συνολικό χρόνο που χρειάστηκε προς τον αριθμό των αιτημάτων. Τυπώνεται το αναγνωριστικό μήνυμα που σημαίνει και το τέλος της εκτέλεσης της διεργασίας. Εδώ ανεβάζουμε και τον sem4. Παρατηρείστε ότι μόνο όταν ο sem4 έχει αυξηθεί τόσες φορές όσες και το πλήθος των διεργασιών-παιδιά, δηλαδή Κ τότε θα σηκώσει τον sem2. Δηλαδή η τελευταία διεργασία αφού ολοκληρώσει, θα δώσει το σήμα στην μητρική διεργασία να εκτελεστεί. Εκεί, θα περάσει τον έλεγχο που διασταυρώνει αν ο sem4 ισούται με Κ, άρα η γονική διεργασία θα περιμένει τα παιδία να τελειώσουν την εκτέλεσή τους και τέλος θα βγει από την while.

Τέλος, απελευθερώνω την μνήμη που χρησιμοποίησαν οι σημαφόροι και αφαιρώ τη διαμοιραζόμενη μνήμη και τερματίζει και η μητρική διεργασία.

Αναμενόμενο output: **->**Διεργασία-παιδί-πελάτης (pid) ζητάει γραμμή i

->Γονική διεργασία-εξυπηρετητής δίνει γραμμή i

->(pid) γραμμή i : . . . (περιεχόμενα γραμμής) . . .

. . .

->--(pid)-- μέσος χρόνος : . . .

Σημείωση: Ενδέχεται κάποια φορά η εκτύπωση του επόμενου αιτήματος να προηγηθεί της εκτύπωσης της απάντησης του προηγούμενου. Αυτό μπορεί να συμβεί διότι είναι αργή η εκτέλεση της printf, ωστόσο θα δείτε πως τα αιτήματα έχουν ικανοποιηθεί με τη σωστή σειρά και λόγω των τοπικών μεταβλητών δεν χάνονται τα σωστά δεδομένα που θέλουμε.