sdi1900036

Εργασία 1η - Λειτουργικά Συστήματα

ΣΗΜΑΝΤΙΚΗ ΣΗΜΕΙΩΣΗ: ΤΟ εκτελέσιμο αρχείο είναι το a.out
Για την υλοποίηση του προγράμματος, αξιοποιήθηκε το POSIX IPC.
Μικρά κομμάτια κώδικα αξιοποιήθηκαν από τα αρχεία του φροντιστηρίου του μαθήματος που έχουν ανέβει στο e-class.

Τα ορίσματα (int argc,char **argv) είναι αναμενόμενα, εφόσον κρίνεται αναγκαία η χρήση πληροφοριών που μας δίνει ο χρήστης μέσω της γραμμής εντολών. Πιο συγκεκριμένα,το πρώτο όρισμα αποτελεί συμβολοσειρά, την οποία χρησιμοποιούμε ως όρισμα σε παρακάτω συναρτήσεις. Το δεύτερο όρισμα, είναι το πλήθος των διεργασιών του παιδιού (k), και ο αριθμός δοσοληψιών κάθε παιδιού (n).

Σε περίπτωση που ο αριθμός των arguments στη γραμμή είναι διαφορετικός του 4, (4 και όχι 3, εφόσον συμπεριλαμβάνεται και το εκτελέσιμο αρχείο, μαζί με το όνομα του αρχείου κειμένου, το k και το n), βγάζουμε μήνυμα λάθους, και σταματάμε την εκτέλεση.

Καθώς τα arguments που προορίζουμε για τα k,n τα λαμβάνουμε από τη γραμμή εντολών, αποτελούν strings. Για να τα αξιοποιήσουμε ως ακέραιους αριθμούς, είναι αναγκαία η χρήση της συνάρτησης atoi.

Η συνάρτηση funct, λαμβάνει ως όρισμα μια συμβολοσειρά (εδώ εισάγουμε το όνομα οποιουδήποτε αρχείου κειμένου επιθυμούμε) , την οποία εκχωρεί στη συνάρτηση fopen ως το όνομα αρχείου, (“r” εφόσον σκοπεύουμε μονάχα να διαβάσουμε το αρχείο, και δεν σκοπεύουμε να το αλλάξουμε με οποιοδήποτε τρόπο). Στόχο της συνάρτησης αποτελεί ο υπολογισμός των αριθμών γραμμών που έχει ένα αρχείο κειμένου, όπου αναφέρεται και στη δεύτερη παράγραφο της εκφώνησης.

Ως shmsize, ορίζω το επιθυμητό μέγεθος της κύριας μνήμης μου, η οποία θα περιέχει έναν ακέραιο, 3 σεμαφόρους για τον σωστό συγχρονισμό των διεργασιών μεταξύ πατέρα και παιδιών, αλλά και μια συμβολοσειρά μεγίστου 101 χαρακτήρων (+1 έτσι ώστε να προσμετρήσουμε και την αλλαγή γραμμής \n). Όπως θα εξηγηθεί και παρακάτω με μεγαλύτερη λεπτομέρεια, οι σεμαφόροι είναι unnamed semaphores, εφόσον χρησιμοποείται η συνάρτηση sem_init σε posix περιβάλλον. Γι’αυτό το λόγο, κρίνεται αναγκαία η ενσωμάτωση τους στη διαμοιραζόμενη μνήμη, για τη σωστή πρόσβαση σε αυτούς από κάθε παιδί διεργασία και τον γονέα του.

Μέσω της shgmet, εκχωρούμε στη μεταβλητή shmid το απαραίτητο identifier με το οποίο θα “ταυτοποιούμε” το συγκεκριμένο memory segment μας, το οποίο όπως προείπαμε, θα είναι μεγέθους SHMSIZE(εξού είναι και όρισμα στη κλήση της συνάρτησης). Ταυτόχεονα, κάνουμε το ανάλογο error checking. Δηλαδή αν το id που προκύπτει για το segment είναι αρνητικό, τότε έχει προκύψει κάποιο λάθος, και κάνουμε exit με κωδικό 1 για να σηματοδοτήσουμε την αιτία της πιθανής αποτυχίας του προγράμματός μας.
Μέσω της shmat, κάνω attach το memory segment που ταυτοποιείται μέσω της μεταβλητής shmid στην υπάρχουσα διεργασία.

Στη συνέχεια δημιουργώ 3 σεμαφόρους, οι οποίοι θα συνδράμουν στον συγχρονισμό της γονικής διεργασίας και των παιδιών της. Συγκεκριμένα, δημιουργώ τους σεμαφόρους read,write και unique, και τους αρχικοποιώ με τις τιμές 0,0 και 1 αντιστοίχως. (Ο λόγος για

την επιλογή αυτών των τιμών αναλύεται σε μεγαλύτερο βάθος παρακάτω.) Οι συγκεκριμένοι σεμαφόροι δημιουργούνται μέσω της συνάρτησης sem_init, της οποίας το αποτέλεσματα εκχωρείται σε ανάλογη μεταβλητή retval. Σε περίπτωση που η συγκεκριμένη μεταβλητή δεν ισούται με το 0, έχει προκύψει σφάλμα κατά την εκτέλεση της συνάρτησης, και γυρνάμε τον αντίστοιχο κωδικό λάθος, τερματίζοντας τη διεργασία.

Κάνω ένα loop, το οποίο διαρκεί k επαναλήψεις, εφόσον όπως δηλώνει και η εκφώνηση K είναι ο αριθμός παιδιών διεργασιών. Μέσα στη συγκεκριμένη επανάληψη, δημιουργώ μέσω fork() παιδιά διεργασίες. Για να γίνει όμως αυτό, πρέπει να σιγουρευτώ οτι αυτά δημιουργούνται από τον server-πατέρα. Σε περίπτωση που το pid (το οποίο εχει το "αποτέλεσμα" της fork()), έχει την τιμή 0, δηλαδή είναι παιδί, σταματάμε επι τόπου την επανάληψη, εφόσον ο στόχος μας είναι η δημιουργία K παιδιών-διεργασιών οι οποίες θα επικοινωνούν με τον γονέα, και όχι να δημιουργούν τα παιδιά και αυτά δικά τους παιδιά.

Στη συνέχεια, σε περίπτωση που βρισκόμαστε σε διεργασία-παιδί, αρχικά σκοπεύουμε στην τυχαία επιλογή μιας συγκεκριμένης γραμμής του αρχείου που δώσαμε ως πρώτο όρισμα στη γραμμή εντολών, θέτωντας πρώτα το seed στην srand.

(Η σειρά περιγραφής των διεργασιών θα ακολουθήσει την φυσική εκτέλεσή τους)

Με μια δομή επανάληψης, η οποία θα διαρκέσει για N επαναλήψεις (αυτός είναι ο αριθμός δοσοληψιών στον οποίον θα εμπλέκεται κάθε παιδί διεργασία, και ο οποίος θα παραμένει σταθέρα N από εκφώνηση), δηλώνουμε μεταβλητές τύπου double (αυτες θα χρησιμεύσουν για την χρονομέτρηση των διεργασιών, και η ακρίβεια του double είναι αναγκαία εφόσον οι διεργασίες εκτελούνται ιδαίτερα γρήγορα. Αρχικά "κατεβάζουμε" το σεμαφόρο unique, μέσω του οποίου σιγουρευόμαστε ότι λειτουργεί αποκλειστικά μια διεργασία παιδί κάθε φορά. Αυτό είναι αναγκαίο καθώς είναι μπορούν να προκληθούν σημαντικά προβλήματα στη shared memory, αν επιχειρήσουν δυο διεργασίες να έχουν πρόσβαση, ή και να την αλλάξουν ταυτοχρόνως. Ο σεμαφόρος unique είναι αρχικοποιημένος με 1, εφόσον σε αντίθετη περίπτωση το πρώτο πρώτο παιδί θα σταμάταγε επι τόπου και θα περίμενε, ενώ τώρα μπορεί να συνεχίσει κανονικά. Λαμβάνω μέσω rand ενα τυχαίο αριθμό από το 1 εώς το lines(εξού και το %lines). Αυτό είναι αναγκαίο, εφόσον αν είχαμε διαλέξει εναν εντελώς τυχαίο αριθμό, θα μπορούσαμε να βγούμε εκτός ορίων του αρχείου κειμένου μας. Θέτω την πρώτη θέση του shared memory (στην οποία έχουμε δηλώσει πιο πάνω οτι θα περιέχει ένα integer),να περιέχει τον rand αριθμό που δημιουργήσαμε, ενώ ταυτόχρονα βάζω τον χρόνο να αρχίζει να μετράει, εφόσον μόλις το παιδί ουσιαστικά υπέβαλε την "αίτηση". Παράλληλα, με το "ανέβασμα" του σεμαφόρου read, ο οποίος αρχικά είναι 0, ενημερώνω τον πατέρα οτι έχω "γράψει" στη shared memory τον αριθμό που επιθυμώ, ενώ ταυτόχρονα μέσω sem_wait στον σεμαφόρο write σταματάω τη λειτουργία του παιδιού (αυτό ισχύει και για την πρώτη πρώτη διεργασία παιδί, εφόσον σεμαφόρος εχει δημιουργηθεί με τιμή 0), έτσι ώστε να αναλάβει δράση ο πατέρας.(οι υπολοιπόμενες εντολές για τις διεργασίες παιδία θα εξηγηθούν μετά την εξήγηση για την λειτουργία του server-πατέρα).

Μέσα σε ένα if, ελέγχουμε αν το τωρινό pid είναι μεγαλύτερο του 0 (στη περίπτωση δηλαδή που είναι ο πατέρας). Μέσα σε δύο εμφωλευμένες δομές επαναλήψεων, η μία που διαρκεί K επαναλήψεις (όσο δηλαδή συνολικά τα παίδια της γονικής διεργασίας), και η εσωτερική N επαναλήψεις, όσο δηλαδή ο σταθερός αριθμός δοσοληψιών του κάθε παιδιού-διεργασίας, μέσω sem_wait στον σεμαφόρο read σταματάει η γονική διεργασία την εκτέλεση της (ακόμη και στην πρώτη περίπτωση, εφόσον η read είναι αρχικοποιημένη με τιμή 0), και περιμένει την αλλαγή της τιμής του σεμαφόρου σε 1. Μόλις αυτό γίνει, δηλαδή γίνει η εντολή sem_post(read) στην child διεργασία, αυτό σημαίνει οτι το παιδί έχει

γράψει τον τυχαίο αριθμό γραμμής που επιθυμεί να παραλάβει στην shared memory. Έτσι,συνεχίζοντας στον γονέα, γράφουμε σε μια μεταβλητή ait, εκχωρούμε το περιεχόμενο της πρώτης θέσης της shared memory (αυτή δηλαδή που περιέχει τον τυχαίο integer που έγραψε στη μνήμη το παιδί).

Μετά, η funct2, η οποία λαμβάνει ως όρισμα μια συμβολοσείρα (η οποία αποτελεί το όνομα του αρχείου κειμένου), ένα ακέραιο num, που αποτελεί και τον αριθμό σειράς που θέλουμε από το αρχείο, και ενα pointer σε integer, που είναι προφανώς ο pointer που δείχνει στη shared memory μας. Μόλις βρούμε μεσω του count, το οποίο αυξάνεται κάθε φορά για κάθε γραμμή του αρχείου, την ζητούμενη γραμμή, την κάνουμε copy ολόκληρη (μέχρι τους 101 χαρακτήρες που απαιτεί η εκφώνηση), στο συγκεκριμένο σημείο που έχουμε δηλώσει στη διαμοιραζόμενη μνήμη, το οποίο βρίσκεται μετα από το integer και τους 3 σεμαφόρους.

Ύστερα,αυξάνω τον σεμαφόρο write, ενημερώνοντας ουσιαστικά το παιδί οτι πλέον η απαιτούμενη γραμμή είναι γραμμένη στο shared memory. Έτσι, ξανά-ενεργοποιείται το παιδί, και επι τόπου σταματάω τη χρονομέτρηση, εφόσον στόχος μας είναι ο υπολογισμός του μέσου χρόνου απο την υποβολή του αιτήματος μέχρι τη λήψη του. Ύστερα, τον χρόνο που υπολόγισμα τον αθροίζω με τους προηγούμενους στη μεταβλητή sumtime. Αυτή μετά το τέλος της εκτέλεσης του παιδιού θα διαιρέσει τη μεταβλητή με τον σταθερό αριθμό δοσοληψιών n, έτσι ώστε να βρούμε τον μέσο χρόνο που έχει προαναφερθεί. Ακόμα, κάνω το παδί για δοκιμή να τυπώσει τη πρόταση που μόλις γράψαμε στη διαμοιραζόμενη μνήμη, ενώ ύστερα αυξάνω το σεμαφόρο unique, δηλώνοντας στις άλλες διεργασίες παιδιά που ενδεχομένως να περιμένουν, να συνεχίσουν την εκτέλεσή τους. Μόλις τελειώσουν όλες οι επαναλήψεις των παιδιών, κάνω detach το συγκεκριμένο segment της shared memory μέσω shmdt.

Τέλος, μέσω της εντολής “while((wpid=wait(&status)>0));” στον πατέρα, η γονική διεργασία αναμένει να τελειώσουν όλες οι διεργασίες παιδιά, πρωτού αυτός προβεί σε detaching κλπ…

Τελικά, μέσω shmctl, και με όρισμα το shmid (που αποτελεί το “αναγνωριστικό” id της διαμοιραζόμενης μνήμης) , και IPC_RMID, αφαιρούμε το identifer shmid, και αφαιρούμε/καταστρέφουμε το αντίστοιχο shared memory segment που του αναλογεί.