Λειτουργικά Συστήματα/Περίοδος 2021-2022
1η Εργασία README

Παναγιώτα Γύφτου, ΑΜ.: 1115201900318

Δεκέμβριος 2021

1.Οι εντολές μεταγλώττισης του προγράμματος βρίσκονται στο αρχείο Makefile, δηλαδή η μεταγλώττιση γίνεται με το ***make***.

2. Και τρέχω το πρόγραμμα για παράδειγμα : ***./parent 5 3 test.txt***
(δηλ. το πλήθος των διεργασιών παιδιών (K) = 5, πλήθος δοσοληψιών= 3, στο tar σας έχω βάλει ένα .txt αρχείο δικό μου)

Έχω δύο processes το parent.c και το client.c.Η επικοινωνία μεταξύ των δύων processes για να καταστεί εφικτή, ώστε η διεργασία parent.c να προωθεί δεδομένα στην διεργασία client.c και αντίστροφα, πρέπει να δεσμευτεί κάποιος ενιαίος χώρος .Αυτός ο χώρος θα είναι η διαμοιραζόμενη μνήμη.Στο parent.c δημιουργούμε την διαμοιραζόμενη μνήμη.Για την δημιουργία της καλώ την shmget και ορίζω ως αναγνωριστικό κλειδί για την ταυτοποίηση της το `(key_t)1234`.Η δημιουργία του κλειδιού έχει γίνει μέσω της μεθόδου casting ενός ακέραιου αριθμού 1234 στον τύπο key_t.Έπειτα ορίζω το μέγεθος της να είναι ίσο με ένα struct `sizeof(struct Shared_Memory)`.Η δομή αυτή περιέχει μια συμβολοσειρά, η οποία θα είναι αυτή που θα δίνει η γονική διεργασία στο παιδί την οποία ζήτησε και ένας ακέραιος αριθμός που θα αντιπροσωπεύει την γραμμή που ζητάει να του σταλεί η διεργασία παιδί.Με την παράμετρο `0666` καθορίζει τα δικαιώματα πρόσβασης στον κατάλογο του αντικειμένου μνήμης κοινής χρήσης.Το τελευταίο όρισμα είναι το `IPC_CREAT` είναι ένας μηχανισμός διεργασιακής επικοινωνίας και θα δημιουργήσει την συγκεκριμένη διαμοιραζόμενη μνήμη αν δεν υπάρχει ήδη.Η shmget επιστρέφει έναν ακέραιο (που το αποθηκεύω στην ακέραια μεταβλητή SH_id) , με τον οποίο θα αναγνωρίζεται μοναδικά η συγκεκριμένη διαμοιραζόμενη μνήμη.Αν η τιμή επιστροφής είναι -1 η δημιουργία απέτυχε και το πρόγραμμα τερματίζεται ενημερώνοντας τον χρήστη.Για να μπορέσει η διεργασία parent.c να χρησιμοποιήσει την διαμοιραζόμενη μνήμη θα κάνουμε attouch μέσω της κλήσης shmat.Η shmat θα κάνει προσάρτηση σε ένα χώρο μνήμης που διαχειρίζεται το συγκεκριμένο process.Η δεύτερη παράμετρος στην shmat είναι ένας pointer που δείχνει σε μια διεύθυνση μνήμης `(void *)0`. Αν η προσάρτηση έγινε επιτυχώς η shmat θα επιστρέψει έναν pointer (shared_memory) που θα δείχνει στο πρώτο byte της διαμοιραζόμενης μνήμης.Αλλιώς το πρόγραμμα τερματίζει.Αντίστοιχα το ίδιο γίνεται και στην διεργασία client.c η shmget καλείται με τις ίδιες παραμέτρους όπως και πριν , με την διαφορά τώρα ότι η διαμοιραζόμενη μνήμη υπάρχει και απλώς της επιστρέφεται ο ακέραιος αριθμός αναγνώρισης (SH_id).Στην συνέχεια για να έχει πρόσβαση με την κλήση shmat, η 2η παράμετρος είναι η ίδια, `(void *)0` , γιατί οι δύο διεργασίες πρέπει να “κοιτάζουν” την ίδια διαμοιραζόμενη μνήμη και της επιστρέφεται ένας pointer που δείχνει στην διαμοιραζόμενη μνήμη.Εάν η shmat αποτύχει τερματίζουμε το πρόγραμμα.

Αποκτάμε πρόσβαση στα περιεχόμενα της διαμοιραζόμενης μνήμης με τον δείκτη shared_mem.Τον οποίο τον δημιουργούμε και στις δύο διεργασίες με την εντολή `shared_mem = (struct Shared_Memory*)shared_memory`

Για να αποφύγουμε το σενάριο της διαπλοκής, με συνέπεια την δημιουργία εσφαλμένων αποτελεσμάτων θα χρησιμοποιήσουμε τον μηχανισμό των σημαφόρων.Οι σημαφόροι που έχω επιλέξει για την υλοποίηση της εργασίας είναι οι POSIX. Στην γονική διεργασία (parent.c) δημιουργώ 3 σημαφόρους.Ο πρώτος σημαφόρος με όνομα, MUTEX_SEM, χρησιμοποιείται από την διεργασία παιδί ώστε να υπάρχει διπλή ασφάλιση,όταν ένα παιδί εκτελεί στην κρίσιμη περιοχή διάφορες ενέργειες, να μην μπει κανένα άλλο.Για την δημιουργία του καλείται η sem_open οι παράμετροι είναι το όνομα, η ένδειξη δημιουργίας *O_CREAT* ,η τρίτη παράμετρος παραπέμπει σε μια δήλωση όπου είναι διάφοροι παράμετροι που αφορούν τα δικαιώματα του σημαφόρου και τέλος αρχικοποιούμε τον σημαφόρο ίσο με το 1.Ο Δεύτερος σημαφόρος που θα συμβάλει στην λειτουργικότητα του προγράμματος είναι ο COUNTER_SEM, ο συγκεκριμένος δυαδικός σημαφόρος δείχνει στο παιδί πότε είναι “άδεια” η διαμοιραζόμενη μνήμη, (δηλ δεν την χρησιμοποιεί ο γονέας) ώστε να μπεί και να εκτελέσει αυτός στην κρισιμη περιοχή.Ο σημαφόρος έχει τις ίδιες παραμέτρους με τον προηγούμενο εκτός από το όνομα(η αρχικοποίηση του είναι και αυτού 1).Ο τελευταίος σημαφόρος έχει το όνομα MESS_SEM και η ιδιότητα του είναι να δείχνει στον γονέα πότε ένα παιδί έγραψε το αίτημα του και περιμένει να του σταλεί η ζητούμενη γραμμή.Ομοίως και αυτός έχει τις ίδιες παραμέτρους εκτός από το όνομα και την τιμή αρχικοποίησης που είναι ίση με 0.Σε περίπτωση που κάποιος σημαφόρος αποτυγχάνει να δημιουργηθεί, δηλαδή η κλήση της sem_open επιστρέψει λάθος το πρόγραμμα τερματίζεται.Για να έχει η διεργασία παιδί και αυτή πρόσβαση στους σημαφόρους για χρήση, με την sem_open δίνει το όνομα του σημαφόρου και έχει μια δεύτερη παράμετρο την *O_RDWR* η οπόια δίχνει ότι γράφει και διαβάζει.Το sem_open είναι το ίδιο για όλους τους σημαφόρους και το μόνο που αλλάζει είναι το όνομα του καθενός.Πάλι σε περίπτωση αποτυχίας της sem_open η εκτελεση του προγράμματος σταματάει και τερματίζει εκτυπώνοντας το κατάλληλο μύνημα ειδοποίησης σφάλματος.

Το διάβασμα του αρχείου το κάνω μέσω από ρεύματα εισόδου στην διεργασία parent.c. Καλώ την fopen να ανοίξει το αρχείο, επιστρέφει ένα ρεύμα με το οποίο θα αναφέρομαι στη συνέχεια σε αυτό το αρχείο που ανοίχτηκε.Εάν επιστρέψει NULL τότε δεν ήταν εφικτό για κάποιο λόγο το άνοιγμα του αρχείου και τερματίζω το πρόγραμμα ενημερώνοντας.Έπειτα διαβάζω σειρά-σειρά όλο το αρχείο μέσω της fgets και μετράω τις γραμμές.Τέλος με την κλήση rewind θέτω το αρχείο από την αρχή για μελλοντική ανάγνωση.

Στη συνέχεια δημιουργώ τα K παιδιά (pid_t pids[K]) , τα οποία τρέχουν στο client.c process.Η γέννηση των παιδιών γίνεται μέσω της fork(), αν η fork επιστρέψει αρνητική τιμή σημαίνει ότι κάτι δεν πήγε σωστά και το πρόγραμμα τερματίζεται .Αλλιώς αν η επιστρεφόμενη τιμή από την fork είναι μηδέν, είναι το παιδί και εκτελεί την execl περνώντας της των αριθμό των γραμμών και το πλήθος των δοσοληψιών του παιδιού με τον γονέα.Η execl υποκαταστεί την εικόνα μνήμης με την διεργασία client.

Στην διεργασία client.c τρέχουμε έναν βρόχο N φορές (N= δοσοληψίες). Μέσα στην επανάληψη αρχικά κάνουμε down τον σημαφόρο COUNTER_SEM μέσω της sem_wait *sem_wait(Counter_Sem)*, αν για κάποιο λόγο η sem_wait επιστρέψει -1 (απέτυχε η sem_wait) το πρόγραμμα τερματίζεται ενημερώνοντας.Αν όλα πήγαν καλά τότε το παιδί που μπήκε συνεχίζει την πορεία του προς την κρίσιμη περιοχή, ο σημαφόρος COUNTER_SEM τώρα έχει την τιμή 0 , όποιος προσπαθήσει να τον κάνει down θα μπλοκαριστεί μέχρι να

ξανά τεθεί ο σημαφόρος ίσο με 1. Πριν φτάσει το παιδί στην κρίσιμη περιοχή γίνεται down του σημαφόρου MUTEX_SEM ,με τον οποίο αν προλάβει το παιδί που ήρθε μετά από το τρέχον και κάνει down το COUNTER_SEM ενώ το τρέχον δεν έχει ολοκληρώσει, τότε ο σημαφορος MUTEX_SEM έρχεται να το φρενάρει μέχρι να ολοκληρώσει το τρέχον παιδί και να κάνει up τον MUTEX_SEM.Στην κρίσιμη περιοχή το παιδί τοποθετεί έναν τυχαίο αριθμό και μετά για να ενημερωθεί ο χρήστης εκτυπώνει ποιό παιδί είναι μέσω της getpid και τον αριθμό της ζητούμενης γραμμής.Στην συνέχεια κάνει up τον σημαφόρο MESS_SEM για να ενημερώσει την διεργασία parent ότι έχει τοποθετήσει και περιμένει να του στείλει την γραμμή `sem_post(Mess_Sem)`.Αυτό που κάνει η διεργασία γονέας είναι σε έναν βρόχο που εκτελείται K*N φορές δηλ θα τρέχει μέχρι να εξυπηρετήσει όλα τα αιματα των K παιδιών.Στην αρχή του βρόχου περιμένει (έχει μπλοκαριστεί) να γίνει up ο σημαφόρος MESS_SEM ώστε να το κάνει down και να μπει στην κρίσιμη περιοχή για να εκτελέσει.Αυτό που κάνει στην κρίσιμη είναι να διαβάσει τον αριθμό γραμμής που έχει ζητήσει το παιδί.Για να βεβαιωθεί ο χρήστης για την εξέλιξη της επικοινωνίας ενημερώνεται με την εκτύπωση ότι είναι ο πατέρας και του ζητήθηκε η γραμμή με αριθμό τάδε.Έπειτα πάλι με τα ρεύματα διαβάζει το αρχείο μέχρι να βρει την γραμμή.Αφού την βρίσκει αποθηκεύει την ζητούμενη γραμμή στην διαμοιραζόμενη μνήμη και αρχικοποιεί ξανά το αρχείο για μελλοντική ανάγνωση με την rewind. Βγαίνει από την κρίσιμη περιοχή και κάνει up τον σημαφόρο COUNTER_SEM.Το παιδί περιμένει να λάβει την τιμή του σημαφόρου COUNTER_SEM μέσω της sem_getvalue, όταν γίνει 1 εκτυπώνεται η γραμμή που στάλθηκε από τον γονέα.

Η εργασία ζητάει η γονική διεργασία να λαμβάνει τους κωδικούς επιστροφής από τα παιδιά που ολοκλήρωσαν , αυτό γίνεται με έναν βρόχο που βρίσκεται στο parent .c (μετά την λούπα που χρησιμοποιείται για την εκτέλεση της κρίσιμης περιοχής,ώστε να γίνουν όλα τα αιτήματα των παιδιών) και με την κλήση της waitpid λαμβάνουμε την επιστρεφόμενη τιμή.Αν επιστραφεί για κάποιο λόγο αρνητική τιμή το πρόγραμμα τερματίζεται.

Ακόμη ζητείται ο μέσος χρόνος των παιδιών.Για να το επιτύχω αυτό έχω βάλει στο client.c , όταν το παιδί μπει στην κρίσιμη περιοχή μια μεταβλητή τύπου double η οποία κρατάει τον χρόνο έναρξης και μια μεταβλητή (ομοίως τύπου double) που να κρατάει τον χρόνο λήψης στο τέλος του αιτήματος και πριν την ανανέωση του βρόχου προσθέτω σε έναν μετρητή την διαφορά τους.Για να βρω το μέσο χρόνο όταν τελειώνει το κάθε παιδί με τις N δοσοληψίες διαιρώ τον μετρητή με το πλήθος των δοσοληψιών (N).Στο τέλος εκτυπώνω τον μέσο χρόνο του παιδιού.

Στο τέλος του client η διεργασία αποπροσαρτάται από την διαμοιραζόμενη μνήμη μέσω της κλήσης shmdt, αν η τιμή επιστροφή της είναι -1, σημαίνει ότι απέτυχε να αποδεσμευτεί από την διαμοιραζόμενη μνήμη. Το ίδιο γίνεται και στην διεργασία parent.Μετά από την αποδεσμευση η διεργασία parent κάνει την αφαίρεση της διαμοιραζόμενης μνήμης μέσω της shmctl

Διαγράφουμε έναν - έναν τους σημαφόρους μέσω της κλήσης sem_unlink αν επιστραφεί αρνητική τιμή η διαγραφή απέτυχε , έτσι τερματίζουμε το πρόγραμμα.Το unlink θα γίνει στην διεργασία που δημιούργησε τους σημαφόρους και αυτή είναι η parent.c