1η ΕΡΓΑΣΙΑ ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Ονοματεπώνυμο: Απόστολος Καρβέλας

A.M.: 1115201800312

ΠΕΡΙΒΑΛΛΟΝ

Ο κώδικας υλοποιήθηκε σε περιβάλλον Ubuntu 20.04 μέσω IDE VSCode και δοκιμάστηκε σε Ubuntu 16.04 μέσω PuTTy.

ΠΕΡΙΕΧΟΜΕΝΑ

Ο φάκελος περιλαμβάνει τα εξής αρχεία:

- README: τεκμηρίωση και τεχνικές λειτουργίες.
- makefile: Αρχείο για την αυτόματη μεταγλώττιση των c αρχείων σε εκτελέσιμα.

Και τα 3 c αρχεία για την υλοποίηση του προγράμματος.

- p1.c
- p2.c
- chan.c

ΕΚΤΕΛΕΣΗ

Για την εκτέλεση του προγράμματος καλούμε την εντολή make που χρησιμοποιεί την makefile για την μεταγλώττιση των αρχείων σε εκτελέσιμα. Στην συνέχεια, τρέχουμε σε 3 terminals τις εντολές ./p1, ./chan και ./p2 αντίστοιχα, με αυτήν την σειρά ώστε να αρχικοποιηθούν οι σημαφόροι και η διαμοιραζόμενη μνήμη από την p1. Μετά την εκτέλεση των προγραμμάτων εμφανίζεται στο πρώτο terminal η υλοποίηση του P1-ENC1, στο δεύτερο τρέχει η CHAN ενώ στο τρίτο η P2-ENC2. Για να τερματιστεί το πρόγραμμα εισάγουμε την συμβολοσειρά term. Ενώ για την αυτόματη διαγραφή των εκτελέσιμων αρχείων μπορούμε να καλέσουμε την εντολή make clean.

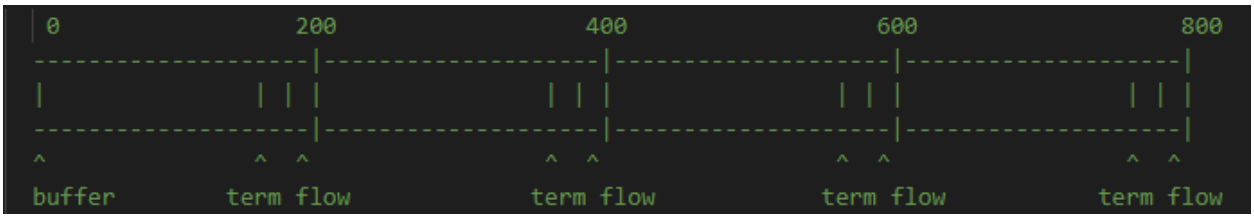
ΥΛΟΠΟΙΗΣΗ ΒΑΣΙΚΩΝ ΣΤΟΙΧΕΙΩΝ

- **ΚΑΤΑΝΟΜΗ ΔΙΕΡΓΑΣΙΩΝ**

Στο πρόγραμμα υπάρχουν 5 διεργασίες, σύμφωνα με την εκφώνηση, η P1/2, η ENC1/2 και τον CHAN. Οι διεργασίες P1 και ENC1 υλοποιούνται στο αρχείο p1.c, οι P2 και ENC2 στο αρχείο p2.c και η CHAN στο chan.c.

- **ΔΙΑΜΟΙΡΑΖΟΜΕΝΗ ΜΝΗΜΗ**

Για το πρόγραμμα χρειάζεται να δημιουργήσουμε κοινή μνήμη μεταξύ των διεργασιών ώστε να μπορούν να συνεννοούνται. Την διαμοιραζομένη μνήμη αυτήν την φτιάχνουμε με την χρήση των εντολών shmget και shmat, οι οποίες περιέχουν το ίδιο κλειδί και η shmget της p1 έχει το ειδικό όρισμα IPC_CREATE| 0666 το οποίο αρχικοποιεί την μνήμη. Ως προς το τεχνικό κομμάτι την μνήμης, δημιουργούμε buffer 800 χαρακτήρων και τον χωρίζουμε στα 4 ίσα κομμάτια. Το πρώτο κομμάτι 200 χαρακτήρων ανήκει στις διεργασίες P1-ENC1, το δεύτερο κομμάτι στις ENC1-CHAN, το τρίτο στις CHAN-ENC2 ενώ το τέταρτο στις ENC2-P2. Με αυτόν τον τρόπο υλοποιούμε ένα πραγματικό σενάριο στο οποίο η μια διεργασία μπορεί να στείλει και να διαβάσει πληροφορίες στις διπλανές διεργασίες. Στους 2 τελευταίους χαρακτήρες κάθε κομμάτι μνήμης υπάρχουν 2 flags αντίστοιχα οι οποίοι βοηθάνε στην πραγματοποίηση της ροής πληροφορίας (flow) και του τερματισμού των διεργασιών (term).



- **ΣΗΜΑΦΟΡΟΙ**

Για την σωστή επικοινωνία των διεργασιών χρησιμοποιούμε σημαφόρους. Στα αρχεία περιέχονται 4 συναρτήσεις για την διαχείριση των σεμαφόρων:

- Sem_create: η οποία φτιάχνει τους τους σημαφόρους με ένα συγκεκριμένο κλειδί ενώ μόνο στο p1 περιέχει το ειδικό όρισμα IPC_CREATE| 0666 στην εσωτερική συνάρτηση semget.
- Sem_init: Αρχικοποιεί τους σημαφόρους με μια τιμή που δέχεται σαν όρισμα.
- P: Η συνάρτηση Wait των σημαφόρων.
- V: Η συνάρτηση Signal.

Στην αρχή κάθε main καλείται η συνάρτηση sem_create(5) για την δημιουργία 5 σεμαφόρων ενώ μόνο στην p1 ύστερα αρχικοποιούνται με 0 μέσω της συνάρτησης sem_init εκτός από τον πρώτο σημαφόρο s0 που γίνεται 1 για να ξεκινήσει το loop, δηλαδή το πρόγραμμα. Στην αρχή κάθε διεργασίας ακριβώς κάτω από το infinity loop υπάρχει μια συνάρτηση P η οποία περιμένει να ενεργοποιηθεί από μια συνάρτηση V που βρίσκεται στο τέλος της προηγούμενης διεργασίας.

- **<u>ΡΟΗ ΠΛΗΡΟΦΟΡΙΑΣ</u>**

Για την υλοποίηση της ροής πληροφορίας χρησιμοποιούμε τον τελευταίο χαρακτήρα από κάθε κομμάτι μνήμης. Όταν το flow ισούται με τον χαρακτήρα '0' σημαίνει ότι υπάρχει δεξιά ροή πληροφορίας (P1→ENC1→CHAN→ENC2→P2) οπότε αρχίζει από την P1 ενώ όταν ισούται με τον χαρακτήρα '1' συμβαίνει το αντίστροιφο, δηλαδή αρχίζει από την P2 (P2→ENC2→CHAN→ENC1→P1). Κάθε διεργασία έχει ένα if…else statement για να καταλαβαίνει την ροή που ακολουθάει η πληροφορία ώστε να αλλάζει το κατάλληλο κομμάτι μνήμης και σημαφόρων. Επίσης, κάθε διεργασία βρίσκεται μέσα σε έναν άπειρο βρόχο (while(1)) ώστε να γίνεται συνεχόμενη ανταλλαγή πληροφοριών.

<u>ΑΡΧΕΙΑ P1 ΚΑΙ P2</u>

Ενώ στο αρχείο chan.c βρίσκεται μόνο η διεργασία CHAN δεν συμβαίνει το ίδιο και στα άλλα 2 αρχεία. Στο αρχείο p1.c υλοποιείται και η διεργασία P1 και η ENC1 όπως και στο αρχείο p2.c υπάρχει και η διεργασία P2 και η ENC2. Οι 2 διεργασίες αυτές μπορούν να συνυπάρχουν σε ένα αρχείο μέσω της συνάρτησης fork() για την παράλληλη εκτέλεση των εργασιών και ενα if…else statement για να καταλαβαίνει που εκτελείται κάθε διεργασία. Αυτή η τεχνική χρειάζεται ώστε να μην είναι κατανεμημένο σε 5 αρχεία και με αυτόν τον τρόπο να γίνεται πιο εύκολη η εκτέλεση και η κατανόηση του προγράμματος.

<u>ΔΙΕΡΓΑΣΙΕΣ P1 ΚΑΙ P2</u>

Στο ξεκίνημα του εκτελέσιμου p1 ο σημαφόρος της P1 αρχικοποιείται με 1 ώστε να αρχίσει πρώτη. Όταν η μεταβλητή flow ισούται με μηδέν, δηλαδή πρέπει να σταλθεί μήνυμα από την P1 στην P2, η P1 διαβάζει την συμβολοσειρά που εισάγουμε και την αποθηκεύει στο πρώτο κομμάτι μνήμης, ύστερα ενεργοποιεί με την συνάρτηση V τον σημαφόρο που βρίσκεται στην αρχή της ENC1. Αν τώρα, το flow ισούται με ένα η P1 εκτυπώνει το μήνυμα που βρίσκεται στο δικό της (αρχικό) κομμάτι μνήμης και αλλάζει στον buffer το flow σε ένα και τέλος ξανατρέχει τον σημαφόρο της P1. Το ίδιο προφανώς συμβαίνει και με την P2 αλλά λειτουργεί αντίστροφα της P1, δηλαδή θα διαβάζει και θα αποθηκεύει ένα μήνυμα στο 4^{ο} κομμάτι μνήμης όταν το flow ισούται με ένα και θα εκτυπώνει, αλλάζοντας στο το flow όταν το flow ισούται με μηδέν.

ΔΙΕΡΓΑΣΙΕΣ ENC1 ΚΑΙ ENC2

Οι διεργασίες ENC1 και ENC2 ξεκινάνε όταν τελειώσει η διεργασία P1 και P2 αντίστοιχα. Όταν το flow ισούται με μηδέν η ENC1 διαβάζει από το πρώτο κομμάτι μνήμης της δοθείσας συμβολοσειράς και την αποθηκεύει στο δεύτερο κομμάτι ώστε να την διαβάσει η διεργασία CHAN, παράλληλα βρίσκει την τιμή κατακερματισμού μέσω της συνάρτησης MD5 και την γράφει δίπλα από την συμβολοσειρά στο 2ο κομμάτι μνήμης, στην συνέχεια ενεργοποιεί τον σημαφόρο της CHAN. Όταν το flow ισούται με ένα διαβάζει την πιθανός αλλοιωμένη συμβολοσειρά από το 2ο κομμάτι μνήμης, δηλαδή από την CHAN, και το checksum (τιμή κατακερματισμού). Ύστερα, υπολογίζει τιμή κατακερματισμού της συμβολοσειράς που μόλις διάβασε και την συγκρίνει με αυτήν που διάβασε από την CHAN (2ο κομμάτι μνήμης). Αν τα checksum είναι ίδια τότε αποθηκεύει την συμβολοσειρά το πρώτο κομμάτι μνήμης και ενεργοποιεί τον σημαφόρο της P1 ώστε να την εκτυπώσει. Ένω αν είναι διαφορετικά καλεί τον σημαφόρο της ENC2 ώστε να ξανασταλθεί η πληροφορία και να ξαναπεράσει αλλοίωση από την CHAN. Με τον ίδιο τρόπο λειτουργεί και η ENC2, απλά αντί για 2ο και 1ο κομμάτι μνήμης διαχειρίζεται το 3ο και 4ο αντίστοιχα και ενεργοποιεί τον σημαφόρο της P2 αν τα checksum είναι ίδια, αλλιώς τον σημαφόρο της ENC1.

ΔΙΕΡΓΑΣΙΑ CHAN

Όταν το flow ισούται με το μηδέν η διεργασία αυτή διαβάζει από την ENC1 (2ο κομμάτι μνήμης) την συμβολοσειρά και την αλλοιώνει. Η αλλοίωση γίνεται παίρνοντας κάθε γράμμα της συμβολοσειράς και το αντικαθιστά με τον χαρακτήρα '*' σύμφωνα με την πιθανότητα που την έχουμε ορίσει με #define PERCENTAGE(η προκαθορισμένη πιθανότητα είναι 5%) στην αρχή του κώδικα. Δηλαδή παίρνει έναν τυχαίο αριθμό από το 0 μέχρι το 99 και αν είναι μικρότερο του PERCENTAGE-1 τότε τον αλλάζει. Στην συνέχεια, διαβάζει το checksum από την ENC1 και γράφει στο 3ο κομμάτι μνήμης (προς το ENC2) την πιθανός αλλοιωμένη συμβολοσειρά μαζί με το checksum και ενεργοποιεί την ENC2. Με την ίδια λογική λειτουργεί και για όταν το flow ισούται με το ένα αλλά όπου ENC1 είναι το ENC2 και αντί για 2ο και 3ο κομμάτι μνήμης ισχύει για 3ο και 2ο .

TERMINATE

Οι διεργασίες βρίσκονται μέσα σε έναν άπειρο βρόχο, οπότε πρέπει να υπάρχει τρόπος τερματισμού του προγράμματος. Όταν η P1/P2 διαβάζει την συμβολοσειρά από τον χρήστη κοιτάει αν αυτή ισούται με "term". Αν τελικά ισχύει τότε αλλάζει στο κομμάτι μνήμης της ENC1/ENC2 τον προτελευταίο χαρακτήρα (term) στον χαρακτήρα '1'. Στην αρχή όλων τον υπόλοιπων διεργασιών τσεκάρει αν το term ισούται με '1' και αν βγει αληθής αλλάζει το term στην ακριβώς διπλανή διεργασία μέχρι να φτάσει στο τέλος

της P2(αν το μήνυμα άρχισε από την P1) ή στο τέλος της P1 (αν το μήνυμα άρχισε από την P2) και χρησιμοποιούν την συνάρτηση exit() ώστε να τερματιστούν. Οπότε όταν οι P1/P2 δουν ότι το term ισούται με ‘1’ διαγράφει όλους τους σημαφόρους όπως και την διαμοιραζόμενη μνήμη.