ΑΜ: 1115202100275

Δημήτριος Χρυσός

Σύντομες Πληροφορίες:

- Έχουν υλοποιηθεί όλα τα ζητούμενα της εκφώνησης
- Το πρόγραμμα αποτελείται από δύο αρχεία (Processes), το «process_a.c» και το «process_b.c». Τα δύο αυτά αρχεία είναι σχεδόν ίδια, με αντίστοιχες ονομασίες μεταβλητών.
- Οι κυριότερες διαφορές μεταξύ τους είναι οι παρακάτω:
    - Το αρχείο «process_a.c» είναι υπεύθυνο για την αρχικοποίηση του συγχρονισμού και της διαμοιραζόμενης μνήμης.
    - Ενώ το αρχείο «process_b.c» λίγο πριν κλείσει, θέτει την ενδιάμεση μνήμη ως έτοιμη προς καταστροφή.

Επεξήγηση του προγράμματος:

- Κάθε ένα από τα δύο αρχεία αποτελείται από δύο structs, δύο συναρτήσεις threads, μία συνάρτηση για εκτύπωση των στατιστικών που ζητούνται στην έξοδο του προγράμματος και την main.
- Το ένα struct «shared_use_st», χρησιμοποιείται για να αποθηκεύει πληροφορίες στο shared memory των δύο διεργασιών.
- Το άλλο struct «statistics», υπάρχει μέσα στο παραπάνω struct, και χρησιμοποιείται για να αποθηκεύει πληροφορίες για την εκτύπωση των στατιστικών στο τέλος.
- Η μία συνάρτηση thread είναι η «send_thread», που χρησιμοποιείται για να παίρνει input από τον χρήστη στο ένα terminal και να στέλνει αυτό το input σε πακέτα των 15 χαρακτήρων στην άλλη διεργασία.
- Η άλλη συνάρτηση thread είναι η «receive_thread», που χρησιμοποιείται για να παραλαμβάνει τα πακέτα της «send_thread», να τα παρασκευάζει σε ένα μήνυμα και να το εκτυπώνει στο άλλο terminal.
- Η συνάρτηση «print_on_exit», χρησιμοποιείται λίγο πριν κλείσει το πρόγραμμα για την εκτύπωση των στατιστικών που ζητούνται.
- Η «main» κάνει τις παρακάτω λειτουργίες:
    - Δημιουργεί ή ανοίγει το Shared Memory.
    - Αρχικοποιεί τις τιμές των μεταβλητών που είναι στο Shared Memory π.χ. semaphores, μεταβλητές για τη λειτουργία του προγράμματος, μεταβλητές για το struct «statistics».
    - Δημιουργεί τα δύο threads, «send_thread – receive_thread», που χρησιμοποιούνται για να στέλνουν και να παρελαμβάνουν μηνύματα για όλη τη διάρκεια που το πρόγραμμα τρέχει.

- Κάνει τα δύο threads, join και κάνει cancel το «send_thread», αν όλα τα άλλα threads και των δύο διεργασιών έχουν κάνει join και το συγκεκριμένο έχει «κολλήσει» στην fgets περιμένοντας input από τον χρήστη.
- Κάνει detach το διαμοιραζόμενη μνήμη και το «process_b» θέτει την ίδια ως έτοιμη προς καταστροφή, η οποία καταστρέφεται αφού κλείσει το πρόγραμμα.

Τεχνικές λεπτομέρειες:

- Γίνεται χρήση Posix Unnamed Semaphores, οι οποίοι αποθηκεύονται στη System V Shared Memory και χρησιμοποιούνται για την ορθή πρόσβαση στην τελευταία.
  Π.χ. μεταξύ των threads για το πότε κάποιο thread είναι σε θέση να κάνει receive ένα μήνυμα ή πρέπει να περιμένει.
  Πιο συγκεκριμένα οι λόγοι που χρησιμοποιείται κάθε Semaphore μέσα στο Shared Memory είναι οι παρακάτω:
  - Τα semaphores «wait_A_receive» στο process_a και «wait _B_receive» στο process_b αντίστοιχα, αρχικοποιούνται με τιμή μηδέν και χρησιμοποιούνται για δύο σκοπούς:
    1. Στο receive_thread λίγο αφού γίνει create το semaphore που ανοίκει στην κάθε διεργασία και μέσα σε ένα while loop, γίνεται sem_wait, άρα το thread περιμένει μέχρι να γίνει sem_post ο semaphore για να συνεχίσει να τρέχει.
       Όταν λοιπόν η άλλη διαδικασία αρχίζει να στέλνει πακέτα ενός μηνύματος κάνει sem_post αυτόν τον semaphore για κάθε πακέτο και άρα η receive αρχίζει να δέχεται πακέτα για να φτιάξει και εκτυπώσει το μήνυμα που της στάλθηκε.
    2. Στην περίπτωση που το input του χρήστη σε μία διαδικασία είναι #BYE# το πρόγραμμα πρέπει να τερματίσει.
       Άρα τότε και μόνο τότε, η send_thread κάνει sem_post το semaphore της receive_thread της ίδιας διαδικασίας, ώστε να συνεχίσει να τρέχει.
       Μετά με ένα έλεγχο που συμβαίνει ακριβώς μετά το sem_wait, βγαίνει από το loop και κάνει pthread_exit ώστε να βγει και από το thread.
  - Τα semaphores «constr_msgA» στο process_a και «constr_msgB» στο process_b αντίστοιχα, αρχικοποιούνται με τιμή μηδέν και χρησιμοποιούνται στην send_thread και receive_thread στη διαδικασία αποστολής και κατασκευής των πακέτων σε μηνύματα.
    Πιο συγκεκριμένα για κάθε πακέτο που στέλνει η μία διαδικασία μετά κάνει sem_wait τον συγκεκριμένο semaphore και περιμένει μέχρι να γίνει post.
    Ο συγκεκριμένος semaphore γίνεται post μόνο στην receive αφού έχει παραλάβει το συγκεκριμένο πακέτο και το έχει συμπεριλάβει στην κατασκευή του μηνύματος που είναι για εκτύπωση.
    Τότε η send_thread συνεχίζει μέσα σε ένα loop να στέλνει το μήνυμα. Αυτή η διαδικασία συνεχίζεται μέχρι να αποσταλεί ολόκληρο το μήνυμα και τότε η send_thread βγαίνει από το εσωτερικό loop και πάει στο εξωτερικό, στο

οποίο είναι έτοιμη να παραλάβει το επόμενο μήνυμα προς αποστολή από τον χρήστη.

- Για την εκτύπωση των στατιστικών χρησιμοποιήθηκαν τα παρακάτω:
    - Το struct «statistics», που αποθηκεύει διάφορες μεταβλητές και «timeval» struct, που αλλάζουν κατά τη διάρκεια του προγράμματος.
    - Την συνάρτηση «print_on_exit», που χρησιμοποιεί τις μεταβλητές από το παραπάνω struct, για να εκτυπώσει τα στατιστικά.
    - Για να βρεθεί ο μέσος χρόνος αναμονής για τη παραλαβή του πρώτου τεμαχίου νέου μηνύματος έγινε χρήση της συνάρτησης «gettimeofday()» και του «struct timeval» της βιβλιοθήκης «<sys/time.h>».
        1. Πιο συγκεκριμένα στην «send_thread», όταν στέλνουμε το πρώτο πακέτο ενός μηνύματος καλούμε την «gettimeofday()» η οποία αποθηκεύει σε ένα struct τύπου «timeval», που υπάρχει μέσα στο struct «statistics» της shared memory, την πληροφορία που δίνει η συνάρτηση.
        2. Ύστερα στην «receive_thread», όταν παραλάβουμε το πρώτο πακέτο του μηνύματος, ξανακαλούμε την «gettimeofday()» η οποία αποθηκεύει σε ένα locally ορισμένο struct τύπου «timeval», την πληροφορία που δίνει η συνάρτηση.
        3. Τότε σε μία μεταβλητή μέσα στο structs «statistics», η οποία είναι αρχικοποιημένη με την τιμή μηδέν, προσθέτουμε την διαφορά μεταξύ των δύο τιμών που δίνουν τα δύο καλέσματα της συνάρτησης «gettimeofday()», σε mircoseconds (δηλαδή την τιμή που δίνει η συνάρτηση στην receive_thread αφαιρόντας την τιμή που δίνει η συνάρτηση στην send_thread).
        4. Αυτό γίνεται για κάθε πρώτο πακέτο ενός μηνύματος
        5. Τέλος παίρνουμε την τιμή που έχουμε φτιάξει στη μεταβλητή που αποθηκεύει κάθε φορά την διαφορά και την διαιρούμε με τα συνολικά μηνύματα που έχει στείλει η άλλη διεργασία σε εμάς.
        6. Αυτό είναι ο μέσος χρόνος αναμονής για τη παραλαβή του πρώτου τεμαχίου νέου μηνύματος.

Αποτελέσματα Προσομοίωσης: