ΔΗΜΗΤΡΙΟΣ ΜΑΛΛΙΔΗΣ
ΑΜ: 1115202300109

ΕΠΕΞΗΓΗΤΙΚΟ ΣΗΜΕΙΩΜΑ ΠΡΩΤΗΣ ΕΡΓΑΣΙΑΣ ΛΕΙΤΟΥΡΓΙΚΑ (README)

Γενικά

Η εργασία αφορά τον σχεδιασμό και την ανάπτυξη ενός συστήματος επικοινωνίας μεταξύ διεργασιών σε περιβάλλον Linux. Ο βασικός στόχος είναι η δημιουργία μιας εφαρμογής που επιτρέπει σε πολλαπλές διεργασίες να ανταλλάσσουν μηνύματα σε πραγματικό χρόνο ταυτόχρονα, χρησιμοποιώντας την τεχνική της διαμοιραζόμενης μνήμης. Το σύστημα υποστηρίζει τη δημιουργία πολλαπλών ανεξάρτητων διαλόγων (conversations/convos), όπου κάθε μήνυμα που αποστέλλεται από μια διεργασία πρέπει να παραλαμβάνεται από όλους τους άλλους συμμετέχοντες στον ίδιο διάλογο ακριβώς μία φορά. Η υλοποίηση βασίζεται στη γλώσσα C.

Αρχιτεκτονική

Η αρχιτεκτονική του συστήματος βασίζεται σε ένα κεντρικό τμήμα διαμοιραζόμενης μνήμης το οποίο χαρτογραφείται στον χώρο διευθύνσεων κάθε διεργασίας που εκκινεί ο χρήστης μέσω της κλήσης mmap. Η δομή της μνήμης ορίζεται στο αρχείο core_protocol.h και οργανώνεται γύρω από τη δομή SharedMemory. Η δομή αυτή περιλαμβάνει έναν πίνακα από convos_limit διαλόγους και έναν πίνακα μηνυμάτων που προκύπτει από το γινόμενο των ορίων διαλόγων και μηνυμάτων ανά διάλογο.

Διαμοιραζόμενη μνήμη (shared memory)

Αυτή η μνήμη περιλαμβάνει τρία τμήματα.

Πρώτον, περιλαμβάνει έναν πίνακα από Convo structs, που αποθηκεύει τα metadata των διαλόγων, όπως το convo_id και το member_count.

Δεύτερον, υπάρχει ένας πίνακας από Message structs, που λειτουργεί ως repository για όλα τα μηνύματα.

Τρίτον, περιέχει έναν καθολικό σημαφόρο shm_mutex (τύπου sem_t) για το synchronization των διεργασιών κατά την πρόσβαση στη μνήμη.

Κάθε Message struct έχει πεδία για το convo_id, το source_id του sender, το sequence_id για το ordering, και έναν reader_count. Ο reader_count είναι κρίσιμος, καθώς καταγράφει πόσες διεργασίες έκαναν read το μήνυμα, επιτρέποντας το αυτόματο cleanup του όταν όλοι οι συμμετέχοντες (member_count) το παραλάβουν.

Συγχρονισμός

Πρέπει να αποφευχθούν καταστάσεις ανταγωνισμού σε ένα περιβάλλον πολλαπλών διεργασιών και νημάτων. Έτσι, χρησιμοποιήθηκαν POSIX Semaphores οι οποίοι αρχικοποιούνται με τη σημαία pshared ίση με 1 μέσω της sem_init, ώστε να είναι ορατοί και λειτουργικοί μεταξύ διαφορετικών διεργασιών.

Η λογική συγχρονισμού χωρίζεται σε δύο επίπεδα.

Στο καθολικό επίπεδο, ο σημαφόρος shm_mutex προστατεύει τη συνολική δομή της μνήμης όταν μια διεργασία αναζητά ένα κενό slot μηνύματος μέσω της συνάρτησης get_available_msg_slot ή όταν προσπαθεί να δημιουργήσει ή να εισέλθει σε έναν διάλογο.

Στο επίπεδο διαλόγου, κάθε Convo struct διαθέτει τον δικό του σημαφόρο (mutex). Αυτός ο σημαφόρος χρησιμοποιείται για την ασφαλή ενημέρωση του αριθμού των μελών (member_count) και την απόδοση μοναδικών sequence IDs στα νέα μηνύματα, διασφαλίζοντας ότι η αύξηση της τιμής next_sequence_id γίνεται ατομικά.

Threads

Κάθε διεργασία δημιουργεί δύο threads μέσω της pthread_create για την επίτευξη ταυτόχρονης λειτουργίας.

Το Sender Thread (process_user_input) διαβάζει το input του χρήστη από το stdin, δεσμεύει ένα ελεύθερο slot στη Shared Memory και καταχωρεί το μήνυμα με το αντίστοιχο sequence_id. Αν ο χρήστης πληκτρολογήσει TERMINATE, το thread θέτει τη σημαία end_check για τον τερματισμό της επικοινωνίας.

Παράλληλα, το Receiver Thread (listen_for_messages) εκτελεί polling στη μνήμη αναζητώντας νέα μηνύματα για το τρέχον convo_id. Όταν εντοπιστεί νέο μήνυμα, εκτυπώνεται στην οθόνη και ενημερώνεται ο reader_count. Αν η διεργασία είναι η τελευταία που διαβάζει το μήνυμα, θέτει το active_check σε false για να ελευθερωθεί ο χώρος.

Είσοδος και Termination

Ξεκινάμε με την κλήση shm_open για τη δημιουργία ή το άνοιγμα του Shared Memory Object. Η πρώτη διεργασία που εκτελείται κάνει το απαραίτητο initialization (ftruncate, memset, sem_init). Ο χρήστης επιλέγει μέσω menu αν θα κάνει create ένα νέο conversation ή αν θα κάνει join σε ένα υπάρχον ID, αυξάνοντας το member_count. Ο τερματισμός (termination) επιτυγχάνεται είτε μέσω του μηνύματος TERMINATE είτε με σήμα SIGINT (Ctrl+C), το οποίο διαχειρίζεται ο signal_handler. Η συνάρτηση shutdown_resources αναλαμβάνει το cleanup, κλείνοντας τους descriptors και καλώντας την shm_unlink μόνο αν δεν υπάρχουν άλλοι ενεργοί διάλογοι στο σύστημα.

Συμπέρασμα

Ο κώδικας μεταγλωττιζεται μέσω Makefile. Για την δοκιμή της εφαρμογής χρησιμοποίησα τα script ανεβασμένα στο eclass αλλά επίσης δούλεψα manually με ssh στα μηχανήματα Linux της σχολής, όπου ανοίγοντας πολλαπλά τερματικά έλεγξα τις απαραίτητες περιπτώσεις (παράλληλα conversations, conversations με πολλά άτομα, TERMINATION με διάφορους τρόπους, κ.λ.π.).