ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Λειτουργικά Συστήματα

Βασιλική Χριστοφιλοπούλου
1115202000216

Χειμερινό Εξάμηνο 2024-2025

Νοέμβριος 2024
Εργασία 1

Για την υλοποίηση του προβλήματος ο κώδικας τμηματοποιήθηκε στα εξής αρχεία:

1. parent.c
2. child.c
3. functions.c
4. functions.h
5. shared_memory.c
6. shared_memory.h

Για να μεταγλωτιστεί η εργασία εκτελούμε:

```
make
./parent <CF.txt> <text.txt> <M>
```

Η εργασία δοκιμάστηκε και τρέχει και για τα τρια configuration files που δώθηκαν.

Σημείωση : Η υλοποίηση μου χρειάζεται η Exit να έχει τρία ορίσματα οπότε να είναι της μορφής 1000 0 EXIT και είναι η μόνη εντολή που τροποποιήθηκε απο τα δοσμένα αρχεία.

Για την μεταγλώττιση των αρχείων έχει δημιουργηθεί και το αντίστοιχο Makefile. Επιπλέον ο κώδικας έχει ελεχθεί μέσω Valgrind για να μην υπάρχουν memory leaks.

```
valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes
--verbose --log-file=valgrind-out.txt ./parent CF1000.txt text.txt 100
```

Ακολουθεί η περιγραφή των αρχείων.

1 parent.c

Πρόκειται για το κύριο τμήμα της εργασίας το οποίο υλοποιεί ότι περιγράφει η εκφώνηση. Αρχικά γίνονται ο έλεγχος των ορισμάτων, το άνοιγμα των αρχείων και οι απαραίτητες αρχικοποιήσεις των μεταβλητών. Στην συνέχεια διαβάζεται μία προς μια η γραμμή του configiration file έως ότου τελειώσει. Για κάθε γραμμή, αποθηκεύονται το timestamp, το process και το command. Κάθε εντολή πρέπει να εκτελείται όταν ο εσωτερικός μετρητής (counter) γίνει ίσος με το timestamp της εντολής. Συνεπώς διακρίνουμε τις ακόλουθες περιπτώσεις:

1. αν counter < timestamp
   Όσο ο μετρήτης είναι μικρότερος απο το timestamp, η διεργασία γονέας ελέγχει ώστε αν υπάρχουν ενεργά παιδιά να διαλέξει ένα στην τύχη και να του αποστείλει μία γραμμή. Προκειμένου να αποστείλει την γραμμή υπάρχουν και οι σημαίες:
   - flag : Αν η σημαία flag είναι 1, τότε σημαίνει ότι είναι σε εξέλιξη μια διαδικασία τερματισμού (terminate), οπότε περιμένει (sem_wait(&sem);) να ολοκληρωθεί η terminate και έπειτα αποστέλει στην γραμμή.
   - spawn : Αν η σημαία spawn είναι 1, αυτό σημαίνει ότι δημιουργείται (fork) νέα διεργασία παιδί. Ο γονέας καλεί pause(), που τον θέτει σε κατάσταση αναμονής, μέχρι το παιδί να στείλει το σήμα SIGCONT, υποδεικνύοντας ότι η δημιουργία ολοκληρώθηκε.
2. αν counter = timestamp
   Εδώ εκτελούνται οι εντολές του αρχείου

- S: Δημιουργία Νέας Διεργασίας
  Εδώ, δημιουργούνται νέες διεργασίες με την χρήση της fork. Αν πρόκειται για διεργασία παιδί, τότε αναζητείται ο πρώτος διαθέσιμος σημαφόρος, αποθηκεύονται οι πληροφορίες στον πίνακα διεργασιών, και εισέρχεται στη συνάρτηση του παιδίου. Ο γονέας ακολουθεί ίδια λογική με την περίπτωση όπου counter < timestamp. Δηλαδή, διαλέγει ένα παιδί στην τύχη και του αποστέλλει το μήνυμα.
- T: Τερματισμός Διεργασίας
  Αρχικά, ελέγχεται αν υπάρχει fork που να εκτελείται. Εφοσον δεν υπάρχει, εντοπίζεται η διεργασία προς τερματισμό με βάση το όνομά της. Αρχικοποιούνται εκνέου τα πεδία που είχε δεσμεύσει, εκτυπώνονται τα σταστιστικά της και ο γονέας αποστέλλει σήμα SIGTERM.
- EXIT: Τερματισμός Όλων των Διεργασιών
  Το πρόγραμμα τερματίζει εκτυπώνοντας τα σταστιστικά των ενεργών διεργασιών και αποδεσμεύοντας την extra μνήμη.

Για τον συχρονισμό μεταξύ γονέα και παιδίου χρησιμοποιούνται οι σημαίες flag, spawn, και οι σημαφόροι message_read (συγχρονίζει την ανάγνωση του μηνύματος), semaphores[i] (συγχρονίζει ότι μπορεί να σταλθεί μήνυμα).

2 child.c

Πρόκειται για την συνάρτηση παιδί, η οποία αρχικά προχωρά σε κάποιες αρχικοποίησεις, αφού μαρκάρει το παιδί ως ενεργό και ενημερώνει τον σημαφόρο ότι είναι έτοιμο να διαβάσει μηνύματα. Ορίζει μια συνάρτηση η οποία θα ενεργοποιείται όταν η διεργασία λαμβάνει signal kill. Έπειτα, όσο είναι ενεργό (active == 1), εισέρχεται στο κρίσιμο τμήμα. Πιο αναλυτικά:

1. Περιμένει μέχρι ο πατέρας να γράψει μήνυμα
2. Διαβάζει το μήνυμα και ενημερώνει τα στατιστικά
3. Αδειάζει τον buffer ώστε να ξέρει ο γονέας ότι ολοκληρώθηκε.
4. Κάνει post τον σημαφόρο ώστε να ενημερωθεί ο πατέρας ότι διάβασε το μήνυμα
5. Επαναλαμβάνεται η ίδια διαδικασία μέχρι active = 0

3 functions.c

Το αρχείο αυτό, περιλαμβάνει όλες τις συναρτήσεις που δημιουργήθηκαν προκειμένου ο κώδικας να τμηματοποιηθεί ωστε να μπορεί να χρησιμοποιηθεί ξανα και να είναι ευανάγνωστος. Πιο αναλυτικά υλοποιήθηκαν οι εξής συναρτήσεις:

1. count_lines : Υπολογίζει τον αριθμό των γραμμών σε ένα αρχείο.
2. find_sem_index : Βρίσκει τον δείκτη σημαφόρου (semaphore index) στον πίνακα διεργασιών βάσει του pid της διεργασίας.
3. find_pid_by_index : Επιστρέφει το PID μιας διεργασίας με βάση τον δείκτη της στον πίνακα διεργασιών.
4. find_pid_by_pid : Επιστρέφει τη θέση στον πίνακα σημαφόρων όπου αντιστοιχεί το συγκεκριμένο pid.
5. find_occurrence : Βρίσκει το PID της nth ενεργής διεργασίας (όχι απαραίτητα στη σειρά του πίνακα).
6. handle_child_termination : Χειρίζεται τον τερματισμό παιδικών διεργασιών όταν λαμβάνουν σήμα SIGTERM.
7. print_stats : Εκτυπώνει στατιστικά στοιχεία για τη διεργασία
8. destroy_semaphores : Καταστρέφει όλους τους σημαφόρους στον πίνακα.
9. count_active_children : Μετράει πόσες διεργασίες στον πίνακα είναι ενεργές

4 functions.h

Προκείται για ένα αρχείο που περιέχει τα δηλώσεις και τα πρότυπα των συναρτήσεων που έχουν υλοποιηθεί στο αρχείο functions.c. Οι δηλώσεις που έχει είναι μεταβλητές οι οποιές χρειάζονται να είναι global.

```
FILE* CF;    // For the configuration file
FILE* text; // For the text file

extern int spawn;
```

5 shared_memory.c

Το αρχείο αυτό περιέχει τις συναρτήσεις που δημιουργήθηκαν για την σύνδεση, την προσκόλληση και την αποσύνδεση μοιραζόμενης μνήμης (shared memory) και την αρχικοποίηση σημαφόρων. Πρόκειται για κώδικα που έχει παρθεί απο το φροντιστήριο και συνγκεκριμένα απο το Semaphores_Lab1 > shm > shmdemo.c. Πιο αναλυτικά υλοποιήθηκαν οι εξής συναρτήσεις:

1. connect_shmem : Δημιουργεί ή αποκτά πρόσβαση σε ένα τμήμα μοιραζόμενης μνήμης.
2. attach_shmem : Προσκολλά την τρέχουσα διεργασία στο τμήμα μοιραζόμενης μνήμης.
3. detach_shmem : Αποσυνδέει τη διεργασία από το τμήμα μοιραζόμενης μνήμης.

6 shared_memory.h

Έδω ορίζονται οι δομές που απαιτούνται για την επικοινωνά μέσω shared memory, καθώς και τα πρότυπα συναρτήσεων που υλοποιήθηκαν και αφορούν την επικοινωνία και τον συγχρονισμό μεταξύ διεργασιών. Καθώς οι πίνακες που υλοποιήθηκαν είναι στατικοί και το μέγεθος τους ορίζεται απο τον χρήστη, στο αρχείο αυτό ορίστηκε ένα αν″ωτατο όριο για τον αριθμό των σημαφόρων MAX_M. Ειδικότερα υλοποιήθηκαν οι δομές:

1. ProcessEntry : Αποθηκεύει βασικές πληροφορίες για κάθε διεργασία και τον αντίστοιχο σημαφόρο:
   - free: 0 αν ο σημαφόρος είναι ελεύθερος, 1 αν είναι κατειλημμένος.
   - pid: Το PID της διεργασίας που χρησιμοποιεί τον σημαφόρο.
   - sem_index: Ο δείκτης του σημαφόρου που χρησιμοποιείται από τη συγκεκριμένη διεργασία.
   - process_index: Ο αριθμός της διεργασίας (π.χ., 1 για την πρώτη διεργασία).
2. ChildInfo : Αποθηκεύει στατιστικά και επιπλέον πληροφορίες για κάθε διεργασία:
   - active: Κατάσταση ενεργότητας (1 = ενεργή, 0 = ανενεργή).
   - received_messages: Αριθμός μηνυμάτων που έχει λάβει η διεργασία.
   - start_timestamp / end_timestamp: Χρονικές στιγμές έναρξης και λήξης της διεργασίας.
   - message: Μήνυμα που γράφεται από τον γονέα για την παιδική διεργασία.
   - my_sem: Ο δείκτης του σημαφόρου που ανήκει στη συγκεκριμένη διεργασία.
   - message_read: Δυαδικός σημαφόρος που συγχρονίζει την ανάγνωση του μηνύματος.
3. shared_data : Αυτή η δομή είναι ο πυρήνας της μοιραζόμενης μνήμης
   - ProcessEntry process_table[MAX_M]: Πίνακας με τις πληροφορίες για όλες τις διεργασίες.
   - sem_t semaphores[MAX_M]: Πίνακας σημαφόρων που χρησιμοποιούνται για συγχρονισμό.
   - ChildInfo childinfo[MAX_M]: Πίνακας με στατιστικά και επιπλέον πληροφορίες για κάθε διεργασία.
   - pid_t child_pids[MAX_M]: Πίνακας για την αποθήκευση των PID των παιδικών διεργασιών.