Λειτουργικά Συστήματα (Κ22) / Περίοδος 2021-2022

1η Εργασία

Μαρία Παρασκευοπούλου

sdi1800155

server.c

Η γονική διεργασία δέχεται ως ορίσματα το αρχείο κειμένου Χ, το πλήθος των διαργασιών Κ και τον αριθμό των δοσοληψιών Ν στις οποίες εμπλέκεται κάθεμια από τις διεργασίες-παιδιά.
Στη γονική διεργασία γίνονται οι εξής ενέργειες:

- Δημιουργία διαμοιραζόμενης μνήμης με την **shmget** η οποία επιστρέφει το identifier του τμήματος διαμοιραζόμενης μνήμης, που αποθηκεύεται στο shmid. Τα ορίσματα που δίνω στην shmget είναι:
  - το **key** είναι **1**
  - το μέγεθος είναι δομής struct row
  - **shmflg** είναι **IPC_CREAT** που δηλώνει δημιουργία shared memory segment αν ένα shmid δεν υπάρχει ήδη για το συγκεκριμένο key.
    Permission 0666: άδεια για γράψιμο και διάβασμα.

  Η **struct row** είναι ορισμένη στο common.h ως:

  ```
  struct row {
      int line_num;
      char text[TEXT_SZ];
      int ready_to_exit;
      double mean_time;
  };
  ```

  όπου θα πάει στο **line_num** ο αριθμός της αιτούμενης γραμμής, στο **text[TEXT_SZ]** η γραμμή με αριθμό line_num (**TEXT_SZ** = 100), στο **ready_to_exit** 1 αν η διεργασία ετοιμάζεται για τερματισμό ή 0 αλλιώς, στο **mean_time** ο μέσος χρόνος από την υποβολή ενός αιτήματος μέχρι την λήψη της αντίστοιχης απάντησης.

- Προσάρτιση του τμήματος διαμοιραζόμενης μνήμης στον χώρο μνήμης της διεργασίας. Κλήση συστήματος **shmat**, η οποία επιστρέφει τη διεύθυνση του attached shared memory segment στην οποία δείχνει ο δείκτης shared_memory.
  ίμι:
  - **shmid** (είχε επιστραφεί από shmget)

  - **shmaddr**, που προσδιορίζει τη διεύθυνση. Εδώ **NULL** άρα το σύστημα επιλέγει μία κατάλληλη αχρησιμοποίητη διεύθυνση για να προσαρτίσει το τμήμα.
  - **shmflg**, εδώ **0**. Η καλούμενη διεργασία (είτε γονέας είτε παιδί) έχει read&write permission άρα το τμήμα διαμοιραζόμενης μνήμης προσαρτείται για διάβασμα και γράψιμο.

  Αν αποτύχει η shmat διαγράφεται το shared memory segment.

- Δημιουργία σετ σημαφόρων με την κλήση συστήματος **semget**. Ορίσματα:
  - το **key** είναι **10**
  - το **semnum**, ο αριθμός των σημαφόρων που δημιουργουνται, είναι 4
  - **flag IPC_CREAT** δηλώνει ότι θέλω δημιουργία
    access permission 0666

  Αν αποτύχει η semget αποσυνδέω και διαγράφω το τμήμα διαμοιραζόμενης μνήμης.

- Οι διαγραφές και αποσύνδεση γίνονται στο

myexit.c

  - **detach_shm(struct row *shared_memory)**
    Καλεί την **shmdt**. Αποσύνδεση του shared memory segment που βρίσκεται στη διεύθυνση shmaddr, εδώ shared_memory, από τον χώρο διευθύσεων που διαχειρίζεται η διεργασία.
  - **remove_shm(int shmid)**
    Καλεί την **shmctl** με ορίσματα:
    * **shmid**
    * **cmd** με τιμή **IPC_RMID**, που δηλώνει καταστροφή του segment.
    * Το **buf** (=**0**) αγνοείται λόγω της τιμής του προηγούμενου ορίσματος.
  - **remove_sem(int semid)**
    Καλεί την **semctl** με ορίσματα:
    * **semid**
    * **semnum**, αριθμός σημαφόρου στο σετ, αγνοείται λόγω της τιμής του τρίτου ορίσματος.
    * **cmd** με τιμή **IPC_RMID**, που δηλώνει διαγραφή ολόκληρου του σετ σημαφόρων με identifier semid.

- Αρχικοποίηση σημαφόρων.

operation.c

  - **init(int semid, int semnum, int value)**
    Αρχικοποίηση του σημαφόρου με αριθμό **semnum** στο σετ με identifier **semid** με την τιμή **value**. Κλήση της **semctl** με flag **SETVAL**.
  - **up(int semid, int semnum)**
    Αύξηση κατά 1 του σημαφόρου με αριθμό **semnum** στο σετ με identifier **semid**. Κλήση **semop** με ορίσματα:

    * **semid**
    * **oper[0]** όπου oper είναι πίνακας που περιέχει ένα στοιχείο τύπου **struct sembuf**, το οποίο πληροφορεί για τον αριθμό του σημαφόρου την τιμή του οποίου θέλω να τροποποιήσω, **semnum**, την πράξη που θέλω να κάνω, εδώ 1 καθώς θέλω να προσθέσω 1, και ένα flag 0 (δεν με ενδιαφέρει).
    * **1**, ο αριθμός των σημαφόρων που θα αυξήσω.
  - **down(int semid, int semnum)**
    Ομοίως με την up.

  Στο πρόγραμμα μου έχω 4 σημαφόρους:

  - **sem1 (semnum=0)**: ελέγχει την πρόσβαση του γονέα στον πόρο. Αρχικοποιείται στο 0.
  - **sem2 (semnum=1)**: ελέγχει την πρόσβασή του κάθε παιδιού στον πόρο, συνδράμει στην οργάνωση των αιτημάτων-αποκρίσεων. Αρχικοποιείται στο 1.
  - **sem3 (semnum=2)**: ¨ειδοποιεί¨ το παιδί που έκανε αίτηση ότι η γραμμή υπάρχει στο τμήμα διαμοιραζόμενης μνήμης. Αρχικοποιείται στο 0.
  - **sem4 (semnum=3)**: ελέγχει την διαδικάσια εξόδου των παιδιών και παραλαβής των κωδικών από γονέα. Αρχικοποιείται στο 0.

- Κλήση **fork()** για γέννηση Κ διεργασιών-παιδιών. Για κάθε παιδί καλείται η **execl** για αντικατάσταση του υπόλοιπου προγράμματος με το executable file **client**. Ορίσματα execl:
  - ολόκληρο το μονοπάτι για το αρχείο
  - string arguments, με πρώτο να είναι το όνομα του αρχείου, μετά ο αριθμός δοσοληψιών N, το πλήθος γραμμών του αρχείου και NULL που δηλώνει το τέλος της λίστας των arguments.
- **Αποτυχία execl.**
  Για Κ επαναλήψεις:
  Η διεργασία-παιδί δημιουργείται, η execl αποτυγχάνει. Η διεργασία-παιδί δηλώνει ότι θα τερματίσει, ο γονέας ελέγχει αν έχει υποβληθεί κάποιο αίτημα, δεν έχει, έτσι βλέποντας ότι η διεργασία τερμάτισε μαζεύει τον κωδικό της.
- **Επιτυχία execl.**

client.c

Μια διεργασία-παιδί δέχεται ώς ορίσματα τον αριθμό N των δοσοληψιών και τον αριθμό των γραμμών του αρχείου κειμένου.

- Κλήση της **shmget** με **key** = 1, **size** struct row και flag **0**, για απλή πρόσβαση και λήψη του id του shared memory segment.
- Με την **shmat** προσαρτώ το τμήμα στον χώρο διευθύνσεων της καλούσας διεργασίας.
- Κλήση **semget** με **key** 10, αριθμός σημαφόρων **4** και flag **0**, απλή πρόσβαση στο σετ σημαφόρων και λήψη του id του.

Ακολουθείται το παρακάτω σενάριο (S : server, C : client):

S: down(sem1) ο γονέας μπλοκάρεται

C: down(sem2) sem2=0, διασφαλίζω ότι η τρέχουσα διεργασία-παιδί θα δουλέψει ανενόχλητη από άλλες διεργασίες-παιδιά που θα θέλουν να υποβάλλουν αιτήματα.
Επιλογή τυχαίου αριθμού γραμμής του αρχείου και γράψιμό του στο διαμοιραζόμενο τμήμα.
up(sem1) sem1=1,ξεμπλοκάρισμα γονέα
down(sem3) το παιδί μπλοκάρεται

S: down(sem1) sem1=0
Γράψιμο γραμμής στο shared memory segment
up(sem3) sem3=1, ξεμπλοκάρισμα διεργασίας παιδιού για παραλαβή γραμμής
down(sem4) blocked, αναμονή για να δηλώσει το παιδί αν θέλει να κάνει exit ή όχι

C: down(sem3) sem3=0
Εκτύπωση γραμμής
Αν έχω κάνει N αιτήματα
    ready_to_exit=1
    up(sem4) sem4=1
    exit(0)
Αλλιώς
    up(sem4) sem4=1
    up(sem2) sem2=1, δίνω σειρά σε άλλη διεργασία-παιδί ή στον εαυτό μου πάλι

S: down(sem4) sem4=0
Αν έχω ready_to_exit=1
    wait()
    ready_to_exit=0
    up(sem2) sem2=1, δίνω σειρά σε άλλη διεργάσια-παιδί αφότου η τελευταία τερμάτισε

Κάθε παιδί επαναλαμβάνει τη διαδικασία έως ότου να έχει ολοκληρώσει N αιτήματα.
Ο γονέας επαναλαμβάνει τη διαδικασία μέχρι να τερματίσουν όλα τα παιδιά.

Makefile

compile: make
run: make run
clean: make clean
file: medusa.txt, text file with 13 lines