SPRAWOZDANIE

SOI - Laboratorium

Semafory

Dominik Szaciłowski

1.Treść zadania

Napisz usluge "chat" dajaca uzytkownikom mozliwosc komunikacji miedzy soba. Zaloz, ze istnieja trzy grupy uzytkownikow tej uslugi: -administratorzy, -uzytkownicy VIP, -zwykli uzytkownicy. Uzytkownicy VIP maja podwyzszone prawa i ich wiadomosci "wskakuja" na poczatek kolejki rozsylanych wiadomosci do pozostalych uzytkownikow. Administratorzy w celach prewencyjnych (np.: po przeczytaniu wulgarnego komunikatu) moga w dowolnie wybranym przez siebie monecie wyczyscic cala kolejke wiadomosci. Zaloz, ze wszyscy uzytkownicy komunikuja sie w wylacznie jednym temacie.

2. Zarys rozwiązania

Producentem w projekcie jest kazdy uzytkownik niezaleznie od tego kim jest. Moga oni wyslac wiadomosc za pomoca funkcji sendmessage(). Wiadomosc ta trafia do bufora. Jesli jest on pelny, uzytkownik czeka w kolecje na to az jakas wiadomosc zostanie usunieta z bufora. Konsumentem jest reader(), ktory rozsyla wiadomosci do kazdego z uzytkownikow. Najpierw sa to admini (dla mozliwosci cenzury wiadomosci), potem reszta uzytkownikow. Moga oni odczytac wiadomosc za pomoca funkcji gotMessage(). Dopiero gdy wszyscy otrzymaja wiadomosc, jest ona kasowana z bufora. Kontrole kolejnosci oraz ochrone przed wystapieniem wyscigow zapewniaja semafory z bilbioteki semaphore.h. Dane pomiedzy procesami wymieniane sa za pomoca shared memory.

3. Utworzone definicje w programie

MAX_MESSAGES – maksymalna ilosc wiadomosci w buforze
MAX_LETTERS – maksymalna liczba znakow w jednej wiadomosci
NAME – unikalna nazwa shared memory, dzieki ktorej mozna uzyskac dostep do chatu
SIZE – rozmiar shared memory
MAX_USERS – maksymalna liczba uzytkownikow w chacie
USER,VIP,ADMIN,EMPTY – dostepne wartosci pola User.who
EMPTY_TEXT – domyslna tresc pustej wiadomosci
ADMIN_MESS–wiadomosc wyswietlana w przypadku usuniecia wiadomosci przez admina
BAD_WORD – slow, ktore aktywuje usuwanie wiadomosci przez admina

4. Utworzone struktury

```
typedef struct User {
   int pid;        - id uzytkownika, w domysle pid procesu aktywujacego uzytkownika
   unsigned who;       - kim jest uzytkownik user,admin,vip, EMPTY dla wartosci pustej
} User;

typedef struct Message {
   char text[MAX_LETTERS];       - tekst wiadomosci
   User user;                    -kto wyslal wiadomosc
} Message;

typedef struct Memory {       - struktura przetrzymujaca wszystkie potrzebne dane do pracy chatu
   Message buffor[MAX_MESSAGES];      //tablica przetrzymujaca wiadomosci
   sem_t full; //semafor do kontroli przepelnienia
   sem_t empty; //semafor do kontroli odczytu z pustego bufora
   sem_t mutex; //zamek dostepu do bufora

   sem_t read[MAX_USERS]; //semafor do powiadomienia uzytkownika o wiadomosci
   sem_t fullUsers; //semafor do kotroli przepelnienia liczby uzytkownikow
   sem_t emptyUsers; //semafor do wstrzymania pracy readera w przypadku braku uzytkownikow
   unsigned numberOfUsers; //aktualna liczba uzytkownikow
   User users[MAX_USERS];  //lista uzytkownikow
```

sem_t waitToDel[MAX_USERS]; //semafor do unikniecia kasowania wiadomosci do momentu odczytania jej przez wszystkich uzytkownikow

}Memory;

5. Utworzone funkcje

Przeznaczenie poszczegolnych funkcji zostalo opisane w kodzie zrodlowym mychat.h (nad kazda z nich).

6. Testy

Czytelnosc testow zostala znacznie poprawiona. Kazdy z nich jest opisany po odpaleniu. Testy zostaly rowniez opisane w kodzie zrodlowym tests.c. Skrypt compile.sh kompiluje kod z odpowiednmi flagami. Nastepnie mozemy odpalic testy za pomoca tests.o.