Kolejki komunikatów IPC MSG

Poznajemy zasady działania kolejek komunikatów funkcje tworzące i usuwające kolejki komunikatów, oraz funkcje wysyłające i odbierające komunikaty.

1.Wstęp teoretyczny

•procesy czytają i piszą do dowolnych kolejek komunikatów
•nie wymaga się by jeden proces czekał już na pojawienie się komunikatu zanim drugi zacznie pisać
•proces może umieścić komunikat w kolejce i zakończyć swoje działanie.
•każdy komunikat w kolejce ma następujące atrybuty
•typ - liczba całkowita typu long
•długość - porcji danych, może być 0
•dane - jeżeli długość jest większa niż 0

•Do tworzenia służy funkcja msgget

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgget (key_t key, int msgflag);
```

funkcja zwróci id kolejki lub -1 gdy się nie powiedzie msgflag określa prawa dostępu

•Do umieszczenia komunikatu w kolejce służy

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgsnd (int msqid, struct msgbuf *ptr, int length, int flag;
```

•ptr jest wskaźnikiem do struktury o następującym wzorcu (template)

```
struct msgbuf
    {
    long mtype; /* typ komunikatu */
    char *mtext; /* dane komunikatu */
    }
```

•mtext jest mylący bo mogą tam być różne dane
•przez wzorzec rozumiemy że ptr musi wskazywać na liczbę całkowitą typu long która zawiera typ komunikatu i poprzedza sam komunikat (o ile długość > 0)
•można zdefiniować własną strukturę np.

```
typedef struct my_msgbuf {
             long mtype;
             short mshort; /*dowolność */
             char mchar[8]; /* dowolność */
             } Message;
```

•lenhgth określa długość komunikatu w bajtach jest to liczba bajtów danych zdefiniowanych przez użytkownika, czyli te po polu long, może być też 0.

•flag może przyjąć stałą symboliczną IPC_NOWAIT lub 0.

•jeżeli będzie IPC_NOWAIT to gdy kolejka jest pełna lub jest zbyt wiele komunikatów w całym systemie to funkcja zwróci natychmiast -1 a errno = EAGAIN.

•gdy msgsnd zakończymy sukcesem dostaniemy zwrotne 0.

•Do odbioru służy

```
int msgrcv(int msqid, struct msgbuf *ptr, int length, long msgtype, int flag);
```

gdzie:

•ptr wskazuje gdzie ma być zapisany komunikat

•length - rozmiar danych ptr'a

•jeżeli flag ustawiona na MSG_NOERROR to jeśli otrzymamy więcej danych niż wskazaliśmy przez length to nadmiar będzie pominięty a funkcja nie wykaże błędu

•msgtype - określa pożądany przez odbiorcę komunikat
•0 oznacza pierwszy który wszedł do kolejki (zasada FIFO)
•>0 pierwszy komunikat którego typ jest taki sam

•<0 pierwszy komunikat z najmniejszych typów które nie są większe niż wartość bezwzględna msgtype.

•flag - określa co należy zrobić gdy w kolejce brak komunikatów

•gdy ustawiony IPC_NOWAIT wtedy natychmiast wracamy, funkcja zwraca -1 i errno = ENOMSG
•jeżeli nie ma IPC_NOWAIT to proces czeka aż nastąpi jedno z 3 zdarzeń
•będzie można otrzymać komunikat żądanego typu
•kolejka komunikatów zostanie usunięta z systemu.
•proces przechwyci odpowiedni sygnał

•jeżeli flag ma MSG_NOERROR zachowuje się jak opisano powyżej.

•Do sterowania kolejką komunikatów służy

int msgctl(int msqid, int cmd, struct msqid_ds. *buff);

Gdy cmd jest:
•IPC_RMID - 0 usunięcie kolejki
•IPC_SET - 1 ustawia
•IPC_STAT - 2 pobiera

2.Do napisania jest gra "kamień, papier, nożyczki".
•gra polega na tym, że 2 lub więcej osób jednocześnie pokazuje, rękami jeden z trzech symboli kamień, papier nożyczki.
Papier zwycięża nad kamieniem bo go zawija, nożyczki zwyciężają nad papierem bo go tną kamień zwycięża nad nożyczkami bo je tępi.
Jeżeli osoba która zwycięży dostaje tyle punktów z iloma przeciwnikami zwyciężyła a zabierane jest tyle punktów z iloma przegrała.
Gdy wszyscy pokażą ten sam symbol jest remis i nikt nie dostaje punktów.
•program składał się będzie z serwera tworzącego a potem usuwającego kolejkę komunikatów oraz klientów chcących grać
•serwer po utworzeniu kolejki co zadany odstęp czasu (np. 5-10s) odczytuje zgłoszenia od klientów (K,N lub P) i przyznaje im punkty.
•jeżeli w wyznaczonym czasie nie ma więcej zgłoszeń niż jedno, serwer przesyła klientowi komunikat, że było za mało zgłoszeń
•jeżeli liczba zgłoszeń jest wystarczająca by zagrać serwer zlicza punkty i rozsyła do wszystkich informacje o bieżącej rozgrywce oraz podsumowanie dotychczasowych rozgrywek
•serwer powinien na bieżąco pokazywać wszystkie niezbędne dane np.: liczba zgłoszeń, kto i jak zagrał ile zgromadził punktów.
•zaprojektować strukturę komunikatu służącą do komunikacji klienci-serwer
•komunikaty typu 1 powinny trafiać do serwera

•komunikaty typu = <pid> powinny trafiać do procesu o podanym identyfikatorze
•klient zgłaszający wysyła komunikat do kolejki komunikatów typu 1 wpisując jednocześnie w odpowiednie pole swój identyfikator i treść zgłoszenia

•serwer wysyła komunikaty o typie = <pid_nadawcy>

•klient powinien odebrać komunikat typu <swoj_pid> i wyświetlić liczbę zdobytych punktów, ile ma obecnie i na którym jest miejscu

•serwer powinien obsłużył sygnał ctrl+c lub i usunąć kolejkę komunikatów.

3.Sprawozdanie
Oczywiście pamiętajcie o sprawozdaniu