**Таблиця А.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Вар.** | **Средство взаимодействия** | **Вар.** | **Средство взаимодействия** |
| 1 | Пам’ять що розділяється | 11 | Пам’ять що розділяється |
| 2 | Спільні файли | 12 | Спільні файли |
| 3 | Черги повідомлень | 13 | Черги повідомлень |
| 4 | Програмні канали | 14 | Програмні канали |
| 5 | Іменовані канали | 15 | Іменовані канали |
| 6 | Пам’ять що розділяється | 16 | Разделяемая память |
| 7 | Спільні файли | 17 | Спільні файли |
| 8 | Черги повідомлень | 18 | Черги повідомлень |
| 9 | Програмні канали | 19 | Програмні канали |
| 10 | Іменовані канали | 20 | Іменовані канали |

Виклик ***shm\_open*** створює і відкриває новий (або вже існуючий) об'єкт пам'яті, що розділяєтья. При відкритті за допомогою функції ***shm\_open()*** повертається дескриптор. Ім'я ***name*** трактується стандартним для розглянутих засобів взаємодії між процесами. За допомогою аргументу ***oflag*** можуть зазначатися прапори **O\_RDONLY**, **O\_RDWR, O\_CREAT, O\_EXCL** та/або **O\_TRUNC**. Якщо об'єкт створюється, то режим доступу до нього формується у відповідності зі значенням ***mode*** і маскою створення файлів процесу. Функція ***shm\_unlink*** виконує зворотну операцію, видаляючи об'єкт, попередньо створений за допомогою ***shm\_open***.

Після підключення сегмента пам'яті що розділяєтья до віртуальної пам'яті процесу цей процес може звертатися до відповідних елементів пам'яті з використанням звичайних машинних команд читання і запису, не вдаючись до використання додаткових системних викликів.

***int main (void) {***

***int fd\_shm; /\* Дескриптор объекта в разделяемой памяти\*/***

***if ((fd\_shm = shm\_open (“myshered.shm”, O\_RDWR | O\_CREAT, 0777)) <0***

***{***

***perror ("error create shm");***

***return (1);***

***}***

Для компіляції програми необхідно підключити бібліотеку ***rt.lib*** у наступний спосіб: ***gcc 1.c-o 1.exe-lrt*** черги повідомлень

Черги повідомлень ***(queue)*** є більш складним методом зв'язку взаємодіючих процесів в порівнянні з програмними каналами. За допомогою черг також можна з одного або кількох завдань незалежним чином посилати повідомлення деякої задачі-приймачу. При цьому тільки процес-приймач може читати і видаляти повідомлення з черги, а процеси-клієнти мають право лише розміщати в чергу свої повідомлення. Черга працює тільки в одному напрямку, якщо необхідний двосторонній зв'язок, слід створити дві черги. Черги повідомлень надають можливість використовувати декілька дисциплін обробки повідомлень:

. **FIFO** - повідомлення, записане першим, буде прочитано першим;

**. LIFO** - повідомлення, записане останнім, буде прочитано першим;

. **пріоритетна** - повідомлення читаються з урахуванням їх пріоритетів;

. **довільний доступ** - можна читати будь-яке повідомлення, а програмний канал забезпечує тільки дисципліну **FIFO.**

Для відкриття черги служить функція ***mq\_open()****,* яка, за аналогією з файлами, створює опис відкритої черги і посилається на нього дескриптором типу ***mqd\_t***, який повертається в якості нормального результату.

***# include <mqueue.h>***

***mqd\_t mq\_open (const char \* name, int oflag, ...);***

При читанні повідомлення з черги воно не видаляється, а може бути прочитана кілька разів. У чергах реально присутні не самі повідомлення, а тільки їх адреси в пам'яті і розміри. Ця інформація розміщується системою в сегменті пам'яті, доступному для всіх задач, які спілкуються за допомогою даної черги. Кожен процес, що використовує чергу, повинен попередньо отримати дозвіл на використання загального сегменту пам'яті за допомогою системних запитів тому, що черга - системний механізм і для роботи з нею потрібні системні ресурси і звернення до самої ОС.

**Сигнали**

З точки зору користувача отримання процесом сигналу виглядає як виникнення переривання. Процес перериває виконання, і управління передається функції-обробнику сигналу. По закінченні обробки сигналу процес може відновити регулярне виконання.

Типи сигналів прийнято задавати спеціальними символьними константами. Системний виклик ***kill ()*** призначений для передачі сигналу одному або декільком специфікувати процесам в рамках повноважень користувача.

**5.2**