**Практическая работа**

**«Разделяемая память»**

**Цели и задачи**

Изучить общие принципы работы с основными средствами межпроцессной коммуникации. Научиться создавать многопроцессные алгоритмы с общей разделяемой памятью. Познакомиться с легковесными процессами – потоками.

**Общие сведения**

**Средства межпроцессной коммуникации (IPC)**

Средствами межпроцессной коммуникации (IPC – Inter-Process Communication) являются сигналы, каналы, сообщения, семафоры, разделяемая память и сокеты. Процессы выполняются в собственном адресном пространстве, они изолированы друг от друга, поэтому необходимы механизмы для взаимодействия процессов, предоставляемые самой операционной системой, и, как правило, расположенные в адресном пространстве системы.

Коммуникация между процессами необходима для решения следующих задач:

1.Передача данных от одного процесса к другому.

2.Совместное использование общих данных несколькими процессами.

3.Синхронизация работы процессов.

Сообщения, семафоры и разделяемую память обобщенно называют System V IPC. Эти механизмы объединяются в единый пакет, потому что их соответствующие системные вызовы обладают близкими интерфейсами, а в их реализации используются многие общие подпрограммы. Вот основные общие свойства всех трех механизмов:

1.Для каждого механизма поддерживается общесистемная таблица, элементы которой описывают всех существующих в данный момент представителей механизма (конкретные сегменты разделяемой памяти, семафоры или очереди сообщений).

2.Элемент таблицы содержит некоторый числовой ключ, который является выбранным пользователем именем представителя соответствующего механизма. Чтобы два или более процесса могли использовать некоторый механизм, они должны заранее договориться об именовании используемого представителя этого механизма.

3.Процесс, желающий начать пользоваться одним из механизмов, обращается к системе с необходимым вызовом, входными параметрами которого является ключ объекта и дополнительные флаги, а ответным параметром является числовой дескриптор, используемый в дальнейших системных вызовах подобно тому, как используется дескриптор файла при работе с файловой системой.

4.Защита доступа к ранее созданным элементам таблицы каждого механизма основывается на тех же принципах, что и защита доступа к файлам.

**Разделяемая память**

Когда процесс А посылает данные другому процессу В через канал, происходят следующие действия: данные копируются из буфера процесса А в буфер ядра, затем эти же данные копируются из буфера ядра в буфер процесса В.Механизм разделяемой памяти позволяет исключить передачу данных через ядро, предоставляя нескольким процессам доступ к одной и той же области памяти – разделяемой памяти.

Необходимо заметить, что разделяемая память имеет и недостаток по сравнению с каналами. Так как работа с разделяемой памятью может осуществляться многими процессами, присутствует вероятность одновременного изменения содержимого памяти несколькими процессами, что может привести к ошибке. Для псевдопараллельных вычислений, вероятность такого события намного меньше, чем при действительных параллельных вычислениях. Но и в этом случае, нельзя не обращать внимания на эту отличительную особенность разделяемой памяти от остальных механизмов коммуникации процессов. Для предотвращения одновременного изменения несколькими процессами разделяемой памяти используются семафоры, с которыми вы познакомитесь, выполняя следующую работу.

Для получения информации по механизмам ipc, используйте команду **$ipcs**.

Для получения информации только по сегментам разделяемой памяти используйте команду **$ipcs –m**.

**Функции для работы с разделяемой памятью**

Для использования функций работы с разделяемой памятью необходимо подключить заголовочные файлы **sys/shm.h** и **sys/ipc.h**.

**int shmget (key\_t key, int size, int shmflag)**

Функция создает новый сегмент разделяемой памяти с ключом key размером size байт, если shmflag равен IPC\_CREAT, или находит существующий сегмент с ключом key, если shmflag равен IPC\_EXCL. Если же shmflag равен IPC\_CREAT | IPC\_EXCL, функция создаст новый сегмент разделяемой памяти с ключом key, только когда не существует другого сегмента разделяемой памяти с тем же ключом. В случае успеха функция возвращает дескриптор разделяемой памяти или отрицательное значение, в случае неудачи. Параметр key может быть равен IPC\_PRIVATE, в этом случае, система сама определяет незанятый ключ для разделяемой памяти.

Параметр shmflag также может включать флаги доступа.

|  |  |
| --- | --- |
| **Флаг доступа** | **Значение флага доступа** |
| 0600 | Разрешить чтение и запись для владельца |
| 0400 | Разрешить чтение для владельца |
| 0200 | Разрешить запись для владельца |
| 0060 | Разрешить чтение и запись для группы |
| 0040 | Разрешить чтение для группы |
| 0020 | Разрешить запись для группы |
| 0006 | Разрешить чтение и запись для всех остальных |
| 0004 | Разрешить чтение для всех остальных |
| 0002 | Разрешить запись всех остальных |

**void \*shmat (int shmid, void \*shmaddr, int shmflag)**

Функция подключает сегмент разделяемой памяти с указанным дескриптором shmid к виртуальной памяти процесса, вызвавшего данную функцию. Все дочерние процессы, созданные процессом-родителем после вызова функции, также будут иметь доступ к разделяемой памяти. Если shmaddr равен NULL, адрес выделяемого сегмента определяется системой, в противном случае shmaddr задает адрес для выделяемого функцией сегмента. Если shmflag равен нулю, разделяемая память будет доступна как для чтения, так и для записи. Если shmflag равен SHM\_RDONLY, разделяемая память будет доступна только для чтения.

**int shmdt (void \*shmaddr)**

Функция отключает сегмент разделяемой памяти с адресом shmaddr от виртуальной памяти процесса, вызвавшего данную функцию.

**Пример работы с разделяемой памятью**

Процесс – родитель создает сегмент разделяемой памяти и порождает дочерний процесс, процесс-родитель находит сумму первой половины элементов массива, процесс-потомок находит сумму второй половины элементов массива и записывает вычисленную сумму в разделяемую память. Родительский процесс дожидается окончания работы потомка, и выводит окончательный результат – сумму всех элементов массива.

При такой организации алгоритма, операции с разделяемой памятью остаются безопасными и при отсутствии семафоров.

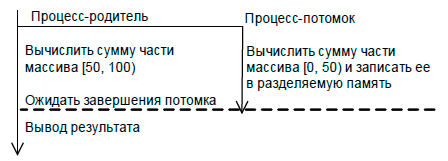


Рис.1

#include <stdio.h>

#include <unistd.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/wait.h>

int A[100] ; *//массив, сумма элементов которого вычисляется процессами*

struct mymem *//структура, под которую будет выделена разделяемая память*

{ int sum ; *//для записи суммы*

} \*mem\_sum ;

int main()

{//запрос на создание разделяемой памяти объемом 2 байта

*//с правами чтения и записи для всех*

int shmid = shmget(IPC\_PRIVATE, 2, IPC\_CREAT|0666) ;

*//если запрос оказался неудачным, завершить выполнение*

if (shmid < 0 ) { fprintf(stdout,"\nОшибка") ; return 0 ; }

*//теперь mem\_sum указывает на выделенную разделяемую память*

mem\_sum = (mymem \*)shmat(shmid,NULL,0) ;

//тут должна быть инициализация элементов массива А

int pid, sum=0 ;

pid = fork() ;

if ( pid == 0 ) *//дочерний процесс*

{

for (int i=0 ; i<50 ; i++) sum+=A[i] ; *//вычислить сумму*

mem\_sum->sum=sum ; *//записать ее в общую память*

}

if ( pid != 0 ) //*родительский процесс*

{

for (int i=50 ; i<100 ; i++) sum+=A[i] ; *//вычислить сумму*

wait(NULL) ; *//дождаться завершения процесса-потомка*

*//вывести на экран сумму всех элементов массива*

fprintf(stdout,"\nРезультат = %d",sum+mem\_sum->sum) ;

}

return 1;

}

**Потоки**

Потоки предоставляют возможность проведения параллельных или псевдопараллельных, в случае одного процессора, вычислений. Потоки могут порождаться во время работы программы, процесса или другого потока. Основное отличие потоков от процессов заключается в том, что различные потоки имеют различные пути выполнения, но при этом пользуются общей памятью. Таким образом, несколько порожденных в программе потоков, могут пользоваться глобальными переменными, и любое изменение данных одним потоком, будет доступно и для всех остальных. Путь выполнения потока задается при его создании, указанием его стартовой функции, созданный поток начинает выполнять команды этой функции, и завершается когда происходит возврат из функции. Любой поток завершается по окончании работы, создавшего его процесса. При создании потока, кроме стартовой функции, ему присуждается буфер для стека, определяемый программистом. Если поток в процессе своей работы превысит размерность стека, выделенного ему программистом, он будет уничтожен системой. Потоки, обладают общей памятью, операции с которой также должны защищаться семафорами.

Для создания потока используется следующая функция (заголовочный файл - sched.h).

**int clone(имя стартовой функции, void \*stack, int flags, void \*arg)**

Функция создает процесс или поток, выполняющий стартовую функцию, стек нового процесса/потока будет храниться в stack, параметр arg определяет входной параметр стартовой функции. Стартовая функция должна иметь такой прототип.

int <имя функции>(void \*<имя параметра>)

Параметр flags может принимать следующие значения.

**CLONE\_VM** – если флаг установлен, создается потомок, обладающий общей памятью с процессом-родителем (поток), если флаг не установлен, создается потомок, которому не доступна память процесса-родителя (процесс).

**CLONE\_FS** - если флаг установлен, потомок обладает той же информацией о файловой системе, что и родитель.

**CLONE\_FILES** - если флаг установлен, потомок обладает теми же файловыми дескрипторами, что и родитель.

**CLONE\_SIGHAND** – если флаг установлен, потомок обладает той же таблицей обработчиков сигналов, что и родитель.

**CLONE\_VFORK** - если флаг установлен, процесс-родитель будет приостановлен до того момента, пока не завершиться созданный им потомок.

Если флаги CLONE\_FS, CLONE\_FILES, CLONE\_SIGHAND, CLONE\_VM не установлены, потомок при создании получает копию соответствующих флагу данных от процесса-родителя. Следует понимать, что копия данных дублирует информацию от родительского процесса в момент создания потомка, но копия и данные, с которых она получена, занимают различные области памяти. Следовательно, изменение данных в процессе работы родителя, неизвестно для потомка, и наоборот.

**Примеры создания потоков**

Процесс-родитель создает поток, выполняющий функцию func.

#include <stdio.h>

#include <sched.h>

#include <unistd.h>

char stack[1000] ; *//для хранения стека потока*

int func(void \* param) *//стартовая функция потока*

{fprintf(stdout, "\nЗапустился поток") ;

return 1 ;

}

int main()

{clone(func,(void\*)(stack+1000-1), CLONE\_VM, NULL) ; *//создать поток*

sleep(1) ; *//заблокируем процесс-родитель, чтобы поток успел выполниться*

return 1 ;

}

Процесс-родитель создает четыре потока, вычисляющих сумму элементов определенной части массива. Созданные потоки, выполняют функцию func, получая в качестве параметра индивидуальное целое число от 0 до 3, с помощью которого, определяются границы вычисления массива каждым потоком.

#include <stdio.h>

#include <sched.h>

#include <unistd.h>

#define NUMSTACK 5000 *//объем стека для отдельного потока*

int A[100] ; *//массив, сумма элементов которого вычисляется процессами*

int SUM=0 ; *//для записи общей суммы*

char stack[4][ NUMSTACK] ; //*//для хранения стека четырех потоков*

int func(void \*param) *//стартовая функция потоков*

{

int i, sum = 0 ; *//для суммирования элементов*

*//индекс массива, с которого начинается суммирование*

int p =\*(int \*)param ; p=р\*25 ;

for(i=p ; i<p+25 ; i++) sum+=A[i] ; *//вычисление суммы части элементов массива*

SUM+=sum ; *//добавление вычисленного результата в общую переменную*

return 1 ;

}

int main()

{//тут должна быть инициализация элементов массива А

int param[4] ; *//для хранения параметров потоков*

for (int i=0 ; i<3 ; i++) *//создание трех потоков*

*{*param[i]=i ; *//каждому потоку передается уникальное число*

char \*tostack=stack[i] ; *//получить указатель на часть массива-стека потоков*

*//создать поток со стартовой функцией func*

*//первый поток получает в качестве параметра 0, второй – 1, третий - 2*

clone(func,(void\*)( tostack+ NUMSTACK -1),CLONE\_VM, (void \*)(param+i)) ;

}

param[3]=3 ; char \*tostack=stack[3] ;

*//создадим четвертый поток, указав процессу дождаться его завершения*

clone(func,(void\*)( tostack+ NUMSTACK -1),

CLONE\_VM|CLONE\_VFORK, (void \*)(param+3)) ;

fprintf (stdout,"\nРезультат = %d", SUM) ;

return 1 ;

}

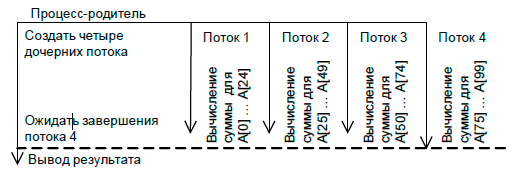


Рисунок 2. Схема потоков

**Порядок выполнения практической работы**

1.Разработать алгоритм решения задания а, с учетом разделения вычислений между несколькими процессами. Для обмена информацией между процессами использовать разделяемую память.

2.Реализовать алгоритм решения задания а и протестировать на нескольких примерах.

3.Разработать алгоритм решения задания б, с учетом разделения вычислений между несколькими потоками.

4.Реализовать алгоритм решения задания б и протестировать на нескольких примерах.

5.Для обоих созданных приложений посмотреть в динамике работу разделяемой памяти, используя команду ipcs -m.

**Варианты заданий**

1.а) Задана строка S, содержащая не менее двух слов. Необходимо найти среди слов, палиндром максимальной длины. Входные данные: строка S. Для решения задачи использовать столько процессов, сколько слов в строке. Палиндромом является фраза или слово, одинаково читаемая как слева направо, так и справа налево, пример – поп.

б) Даны результаты сдачи экзамена по группам. Создать многопоточное приложение, вычисляющее количество двоечников и отличников в каждой группе.

2. а) Найти максимальный M и минимальный элемент m массива А и составить множество чисел, лежащих в интервале (m, M) и не содержащихся в массиве А. Входные данные: целое положительное число n, массив чисел А размерности n.

б) Дан список студентов по группам. Создать многопоточное приложение для определения количества студентов с фамилией Иванов.

3. а) Определить какая сумма больше, сумма всех положительных чисел, стоящих выше главной диагонали и ниже второй главной диагонали, или сумма модулей всех отрицательных чисел, стоящих выше второй главной диагонали и ниже главной диагонали. Входные данные: целое положительное число n, массив чисел А размерности nxn.

б) Охранное агентство разработало новую систему управления электронными замками. Для открытия двери клиент обязан произнести произвольную фразу из 10 слов. В этой фразе должно встречаться заранее оговоренное слово, причем только два раза. Создать многопоточное приложение, управляющее замком.

4. а) Найти среднее арифметическое всех «особых» элементов матрицы А. Будем считать, элемент особым, если он больше суммы всех остальных элементов, стоящих в том же столбце. Входные данные: целое положительное число n, массив чисел А размерности nxn. Использовать n или n+1 процессов для решения задачи.

б) Даны результаты сдачи экзамена по группам. Создать многопоточное приложение, вычисляющее общий средний балл и средний балл для каждой группы.

5. а) Найти сумму индекса строки с максимальным элементом на главной диагонали с индексом столбца с минимальным элементом на второй главной диагонали. Входные данные: целое положительное число n, массив чисел А размерности nxn.

б) Командиру воинской части полковнику Кузнецову требуется перемножить два секретных числа. Полковник Кузнецов вызывает дежурного по части лейтенанта Смирнова и требует предоставить ему ответ. Лейтенант Смирнов будит старшего по караулу сержанта Петрова и приказывает ему предоставить ответ. Сержант Петров вызывает к себе рядового Иванова, и поручает ему ответственное задание по определению произведения. Рядовой Иванов успешно справляется с поставленной задачей, и ответ передается полковнику Кузнецову. Создать многопоточное приложение, в котором все военнослужащие от полковника до рядового моделируются потоками одного вида.

6. а) Составит строку из максимальных и минимальных элементов строк матрицы А. Порядок элементов в строке не важен. Входные данные: целое положительное число n, целое положительное число к, массив чисел от 0 до 9 А размерности nxк. Использовать n или n+1 процессов для решения задачи.

б) Изготовление знаменитого самурайского меча – катаны происходит в три этапа. Младший ученик мастера выковывает заготовку будущего меча. Затем старший ученик мастера закаливает меч в трех водах – кипящей, студеной и теплой. И в конце мастер собственноручно изготавливает рукоять меча и наносит узоры. Создать многопоточное приложение, в котором мастер и его ученики представлены разными потоками.

7. а) Определить количество чисел m, являющихся квадратами некоторого целого числа, в матрице А. Заменить все простые числа в А на m. Входные данные: целое положительное число n, массив чисел А размерности nxn.

б) Дана последовательность натуральных чисел a0…an–1. Создать многопоточное приложение для поиска суммы Σai, где ai– четные числа.

8. а) Найти сумму индексов всех седловых точек матрицы А. Будем считать, элемент седловой точкой, если он является наименьшим в своей строке и наибольшим в своем столбце, либо наоборот. Входные данные: целое положительное число n, массив чисел А размерности nxn.

б) Дана последовательность натуральных чисел a0…an–1. Создать многопоточное приложение для вычисления выражения a0-а1+a2-а3+a4-а5+...

9. а) Найти максимальный и минимальный среди тех элементов матрицы А, сумма индексов которых равна двойке в любой целочисленной степени. Входные данные: целое положительное число n, целое положительное число к, массив чисел от А размерности nxк.

б) Дана последовательность натуральных чисел a0…an–1. Создать многопоточное приложение для поиска всех ai, являющихся квадратами, любого натурального числа.

10. а) Определить является ли матрица А симметричной относительно главной диагонали. Входные данные: целое положительное число n, массив чисел А размерности nxn. Использовать не менее 4 процессов для решения задачи.

б) Дана последовательность натуральных чисел a0…an–1. Создать многопоточное приложение для поиска всех ai, являющихся простыми числами.

11. а) В матрице А найти в каждой строке наибольший элемент и поменять его местами с элементом, стоящим на главной диагонали и в той же строке. Входные данные: целое положительное число n, массив чисел А размерности nxn. Использовать n или n+1 процессов для решения задачи.

б) Дана последовательность натуральных чисел a0…an–1. Создать многопоточное приложение для поиска минимального ai.

12. а) Упорядочить по возрастанию элементы в каждой строке матрицы А. Входные данные: целое положительное число n, целое положительное число к, массив чисел от А размерности nxк. Использовать n или n+1 процессов для решения задачи.

б) Дана последовательность натуральных чисел a0…an–1. Создать многопоточное приложение для поиска произведения чисел a0\*а1\*…\*an–1.

13. а) В массиве строк хранятся фамилии и оценки учащихся. Найти учащегося с максимальным средним баллом, вывести список всех учащихся, имеющих однофамильцев. Входные данные: целое положительное число n, массив строк размерности n. Использовать n или n+1 процессов для решения задачи.

б) Дана последовательность символов С = {с0…сn–1} и символ b. Создать многопоточное приложение для определения количество вхождений символа b в строку C.

14. а) Найти произведение всех элементов в матрице А, сумма или разность индексов которых является простым числом, отрицательные разности не рассматривать. Входные данные: целое положительное число n, целое положительное число к, массив чисел от А размерности nxк.

б) Дана последовательность символов С = {с0…сn–1}. Дан набор из N пар кодирующих символов (ai,bi). Создать многопоточное приложение, кодирующее строку С следующим образом: строка разделяется на подстроки и каждый поток осуществляет кодирование своей подстроки.

15. а) Вычислить произведение матрицы А на В, где матрица В получена из матрицы А, заменой отрицательных элементов нулем и последующим транспонированием. . Входные данные: целое положительное число n, массив чисел А размерности nxn.

б) Дана последовательность натуральных чисел a0…an–1. Создать многопоточное приложение для поиска суммы квадратов Σai2.

16. а) Двоичные числа записаны в строке, разделителем является пробел. Количество чисел равно m. Найти сумму всех двоичных чисел как двоичное число и как десятичное число. Входные данные: строка S. Для решения задачи использовать не менее m процессов.

б) Дана последовательность символов С = {с0…сn–1}. Дан набор из N пар кодирующих символов (ai,bi). Создать многопоточное приложение, кодирующее строку С следующим образом: поток 0 заменяет в строке C все символы a0на символы b0, поток 1 заменяет в строке C все символы a1на символы b1, и т.д. Потоки должны осуществлять кодирование последовательно.

17. а) Проверить, можно ли составить слово S из элементов символьного массива С. Учитывать количество требуемых символов для составления слова. Входные данные: строка S, массив символов С. Использовать для решения задачи столько процессов, сколько неповторяющихся символов в строке S.

б) Даны последовательности символов А = {а0…аn–1} и С = {с0…ск–1}. В общем случае n ≠ k. Создать многопоточное приложение, определяющее, совпадают ли посимвольно строки А и С.

18. а) Строка содержит произвольный русский текст. Вывести сколько раз в ней встречается буква а, буква б, буква в и т.д. Найти три наиболее часто встречающиеся буквы. Входные данные: строка S. Использовать для решения задачи столько процессов, сколько букв в русском алфавите.

б) Дана квадратная матрица А. Создать многопоточное приложение для поиска сумм строк и столбцов.

19. а) Найти максимальный по модулю элемент в матрице А и его индексы – х, у. Поменять местами строку х со строкой к и столбец у со столбцом к. Входные данные: целое положительное число n, массив чисел А размерности nxn, целое положительное число к ≥ 0 и ≤ n-1.

б) Дана последовательность натуральных чисел a0…an–1. Создать многопоточное приложение для поиска максимального ai.

20. а) В массиве А заменить отрицательные элементы нулями, а положительные элементы единицами. Перевести полученное двоичное число в десятичную систему счисления. Входные данные: целое положительное число n, массив чисел А размерности n, целое положительное число к ≥ 2 и ≤ n/2. Использовать для решения задачи к процессов.

б) Изготовление знаменитого самурайского меча – катаны происходит в три этапа. Младший ученик мастера выковывает заготовку будущего меча. Затем старший ученик мастера закаливает меч в трех водах – кипящей, студеной и теплой. И в конце мастер собственноручно изготавливает рукоять меча и наносит узоры. Требуется создать многопоточное приложение, в котором мастер и его ученики представлены одинаковыми потоками (обработка производится в цикле).