Submit a solution for up15-1-mz15-1 (дореш)

Full score:	100
Run penalty:	10
Time limit:	2 s
Real time limit:	2 s
Memory limit:	64M
Stack limit:	8M
Open date:	2022-12-07 17:00:00
Deadline:	2022-12-19 19:00:00
Date penalty formula:	2022/12/31 0-50/7d
Next soft deadline:	2022-12-14 17:00:00

Problem up15-1: mz15-1 (дореш)

Программе в аргументах командной строки задаются:

- Количество процессов (nproc) (1 <= nproc <= 400);
- Ключ IPC (key);
- Максимальное значение (maxval) (32-битное знаковое целое положительное значение).

Программа должна создать nproc процессов и запустить между ними пересылку числа. Процессы нумеруются от 1 до nproc. Процесс-родитель имеет номер 0. Каждый процесс получает число, печатает свой порядковый номер, значение числа, и номер процесса-отправителя как показано в примере. Затем процесс увеличивает его на 1 и пересылает число процессу с номером (val * val * val * val) % nproc + 1, где val – значение числа после увеличения. Если значение числа после увеличения стало больше `maxval`, процесс не пересылает число следующему, а (совместно с родителем) выполняет действия по завершению игры. Игра начинается с первого из созданных процессов со значения 0. Отец дожидается завершения всех процессов и сам завершает работу.

Обратите внимание, при достижении максимального значения оно печатается, но не пересылается дальше.

Можете использовать следующие варианты создания процессов и синхронизации:

- Процессы, SysV семафоры (sys/sem), SysV разделяемая память (sys/shm).
- Процессы, SysV семафоры, mmap.
- Процессы, eventfd, mmap.
- Процессы, POSIX semaphore, mmap.
- Нити, mutex, condvar.

Второй аргумент командной строки не используется, если программа не использует SysV IPC.

Другие варианты использовать запрещено.

Объекты, созданные для работы программы, например, массив семафоров, должны быть уничтожены.

Тестирование завершается с вердиктом 'Synchronization error', если процесс-отец (то есть ваша программа, запускаемая на тестирование) заканчивает работу раньше какого-либо из своих потомков.

Тестирование завершается с вердиктом 'Security violation', если после завершения работы вашей программы остались неудаленные объекты IPC.

Examples

Input

```
```

Output

```
1 0 0
2 1 1
1 2 2
2 3 1
1 4 2
```

Submit a solution

Language: gcc-32 - GNU C (32 bit) 12.1.1

File: Выберите файл | Файл не выбран

Send! Send!

View
all / necessary

up02-4
up02-5
up02-6
up02-7
mz09-1
mz09-2
mz09-3
mz09-4
mz09-5
mz09-6
mz15-1
up15-1
up15-2
up15-3
up15-4
up15-5

Submit a solution for up15-2-mz15-2 (дореш)

Full score:	100
Run penalty:	10
Time limit:	2 s
Real time limit:	4 s
Memory limit:	64M
Stack limit:	8M
Open date:	2022-12-07 17:00:00
Deadline:	2022-12-19 19:00:00
Date penalty formula:	2022/12/31 0-50/7d
Next soft deadline:	2022-12-14 17:00:00

Problem up15-2: mz15-2 (дореш)

Программе в аргументах командной строки передается имя произвольного файла.

Главный процесс создает двух сыновей. Второй сын пересылает первому сыну содержимое указанного файла, а первый сын выводит полученные данные на стандартный поток вывода.

Для пересылки файла используется механизм сигналов: файл пересылается побитно, для пересылки бита 0 второй сын отправляет первому сыну сигнал SIGUSR1, а для пересылки бита 1 - сигнал SIGUSR2. Первый процесс подтверждает получение каждого бита с помощью сигнала SIGALRM. После завершения пересылки всего файла второй процесс посылает первому процессу сигнал SIGIO.

Главный процесс должен дождаться завершения работы обоих процессов и завершиться сам.

Использовать другие средства межпроцессного взаимодействия запрещено.

Submit a solution

Language: gcc-32 - GNU C (32 bit) 12.1.1

File Выберите файл Файл не выбран

Send! Send!

View
all / necessary

up02-4
up02-5
up02-6
up02-7
mz09-1
mz09-2
mz09-3
mz09-4
mz09-5
mz09-6
mz15-2
up15-1
up15-2
up15-3
up15-4
up15-5

Submit a solution for up15-3-mz15-3 (дореш)

Full score:	100
Run penalty:	10
Time limit:	1 s
Real time limit:	2 s
Memory limit:	64M
Stack limit:	8M
Open date:	2022-12-07 17:00:00
Deadline:	2022-12-19 19:00:00
Date penalty formula:	2022/12/31 0-50/7d
Next soft deadline:	2022-12-14 17:00:00

Problem up15-3: mz15-3 (дореш)

В аргументах командной строки программе задается одно целое число `count` - число процессов-сыновей (1 ≤ count ≤ 100).

На стандартном потоке ввода подается последовательность 32-битных знаковых целых чисел в текстовом виде. Последовательность завершается признаком конца ввода.

Родитель создает `count` сыновей, которые нумеруются от 0 до count-1. Сыновья работают последовательно. В свою очередь каждый сын считывает одно число из входной последовательности, выводит на стандартный поток вывода свой номер и считанное число. Затем очередь работы передается сыну с номером, равным остатку (математическому) от деления считанного числа на количество процессов.

После создания всех процессов родитель передает очередь чтения процессу с номером 0. Затем родитель дожидается завершения работы сыновей и сам завершает работу.

Можете использовать следующие варианты создания процессов и синхронизации:

- Процессы, SysV семафоры (sys/sem).
- Процессы, POSIX semaphore.
- Процессы, eventfd.
- Нити, mutex, condvar.

Другие варианты использовать запрещено.

Идентификатор массива семафоров выбирайте самостоятельно. В конце работы программы все созданные объекты IPC должны быть уничтожены.

Стандартные потоки ввода и вывода являются каналами.

Ниже приведен пример работы при числе процессов 3.

Examples

Input

```
1 2 4 3 5 6 9 2
```

Output

```
0 1
1 2
2 4
1 3
0 5
2 6
0 9
0 2
```

Submit a solution

Language: gcc-32 - GNU C (32 bit) 12.1.1

File Выберите файл Файл не выбран

Send! Send!

View
all / necessary

up02-4
up02-5
up02-6
up02-7
mz09-1
mz09-2
mz09-3
mz09-4
mz09-5
mz09-6
up15-1
up15-2
up15-3
up15-4
up15-5
up16-1
up16-2
up16-5

Submit a solution for up15-4-mz15-4 (дореш)

Full score:	100
Run penalty:	10
Time limit:	1 s
Real time limit:	2 s
Memory limit:	64M
Stack limit:	8M
Open date:	2022-12-07 17:00:00
Deadline:	2022-12-19 19:00:00
Date penalty formula:	2022/12/31 0-50/7d
Next soft deadline:	2022-12-14 17:00:00

Problem up15-4: mz15-4 (дореш)

Программе задаются следующие аргументы командной строки:

- Ключ IPC для очереди сообщений
- Число процессов для запуска n
- 64-битное целое число value1
- 64-битное целое число value2
- 64-битное целое число maxval

Программа должна создать n процессов. Каждый процесс имеет свой номер от 0 до n - 1. Когда процесс получает очередь хода, с помощью очереди сообщений процесс должен получить два 64-битных целых числа x1, x2, вычислить их сумму x3 = x1 + x2, напечатать на стандартный поток вывода свой номер и число x3, затем переслать числа x2 и x3 процессу с номером x3 % n.

Гарантируется, что x3 всегда будет неотрицательным.

Если x3 по модулю больше значения maxval, то вместо пересылки следующему процессу все процессы должны завершиться.

В начале игры родительский процесс посылает процессу 0 числа value1 и value2.

Родительский процесс должен дождаться завершения работы всех процессов и сам завершиться.

Родительский процесс должен корректно обрабатывать ошибочное завершение fork(). При этом все уже созданные процессы должны быть уничтожены. Родитель должен завершиться с кодом 1. Все процессы должны использовать одну очередь сообщений.

Можете использовать следующие варианты создания процессов и синхронизации:

- Процессы, SysV очередь сообщений (sys/msg).
- Процессы, POSIX очередь сообщений.
- Нити, mutex, condvar. condvar один на все нити.

Первый аргумент командной строки не используется, если программа не использует SysV IPC.

Другие варианты использовать запрещено.

Объекты, созданные для работы программы, например, массив семафоров, должны быть уничтожены.

Пример запуска программы:

```
../solution 1234 2 0 1 15
```

Результат работы:

```
0 1
1 2
0 3
1 5
1 8
0 13
1 21
```

Submit a solution

Language: gcc-32 - GNU C (32 bit) 12.1.1

File Выберите файл Файл не выбран

Send! Send!

View
all / necessary

up02-4
up02-5
up02-6
up02-7
mz09-1
mz09-2
mz09-3
mz09-4
mz09-5
mz09-6
mz15-4
up15-1
up15-2
up15-3
up15-4
up15-5

Problem up15-5-mz15-5 (дореш)

Full score:	100
Run penalty:	10
Time limit:	1 s
Real time limit:	2 s
Memory limit:	64M
Stack limit:	8M
Open date:	2022-12-07 17:00:00
Deadline:	2022-12-19 19:00:00

Problem up15-5: mz15-5 (дореш)

Программе в аргументах командной строки передаются имена файлов (от 0 до 20 включительно). Файлы представляют собой именованные каналы, по которым программе будут передаваться данные блоками по 16 байт. То есть программа должна открывать эти файлы только на чтение и считывать данные из них блоками по 16 байт.

Программа должна напечатать на стандартный поток вывода свой PID и перейти в режим ожидания поступления сигналов.

При поступлении сигнала SIGTERM программа должна для каждого файла распечатать сумму чисел, считанных из него, и завершить работу. Каждое число выводится на отдельной строке стандартного потока вывода. Если из файла не было считано чисел, их сумма равна 0.

При поступлении сигнала SIGRTMIN + x программа должна переключиться в режим чтения данных из входного файла с индексом x. То есть, например, сигнал SIGRTMIN + 2 переключает программу в режим чтения из файла с индексом 2, путь к которому задается в аргументе командной строки argv[3]. Если этот файл уже закрыт, то программа переходит в режим ожидания поступления сигнала.

В режиме чтения входного файла программа считывает данные из текущего файла блоками по 16 байт. Каждый блок содержит пробельные символы и текстовое представление 64-битного целого знакового числа. Символ с индексом 15 в блоке данных является символом \n. Считанное число прибавляется к текущей сумме чисел этого файла. Гарантируется, что для всех вычислений достаточно 64-битного знакового целого типа. При достижении конца файла он закрывается, а программа переходит в режим ожидания поступления сигнала.

Кроме того, в режиме чтения входного файла обрабатываются сигналы SIGTERM и SIGRTMIN + x как описано выше.

Обработчики сигналов должны не выполнять никаких действий, кроме изменения значений глобальных переменных. Допускается не более 3 глобальных переменных целого типа.

Допускаются следующие варианты решения:

- С обработкой сигналов при отключенном SA_RESTART. В этом случае допускается race condition около системного вызова read.
- С использованием select/pselect/poll/ppoll/epoll.

Использовать неблокирующий ввод-вывод, асинхронный ввод-вывода, создавать новые процессы или нити запрещено.

View
all / necessary

up02-4
up02-5
up02-6
up02-7
mz09-1
mz09-2
mz09-3
mz09-4
mz09-5
mz09-6
mz15-5
mz15-5
up15-1
up15-2
up15-3
up15-4
up15-5