Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра ЭВМ

Дисциплина: Операционные системы и системное программирование

ОТЧЁТ
к лабораторной работе №4
на тему
Взаимодействие и синхронизация процессов.

Выполнил студент гр.230501 Кочеров Р.С.

Проверил старший преподаватель кафедры ЭВМ
Поденок Л.П.

Минск 2024

1 ОПИСАНИЕ АЛГОРИТМОВ И РЕШЕНИЙ.

Данная программа представляет собой простую систему обмена сообщениями между процессами в операционной системе Linux, реализованную с использованием разделяемой памяти и семафоров. Программа создает процессы-производители (продюсеры), которые создают сообщения и помещают их в очередь сообщений, и процессы-потребители (консьюмеры), которые извлекают сообщения из очереди и обрабатывают их.

Вот основные компоненты и функции программы:

Структуры и глобальные переменные: Программа определяет структуры для сообщений и очереди сообщений, а также объявляет глобальные переменные для семафоров, массивов производителей и потребителей.

Инициализация: В функции `initializeQueue()` происходит инициализация разделяемой памяти, семафоров и других необходимых элементов.

Функции создания и удаления производителей и потребителей: Есть функции `createProducerThreads(prodCount)` и `createConsumerThreads(consCount)`, которые создают новые процессы-продюсеры и процессы-потребители соответственно. Аналогично, есть функции `destroyThread(prodIdMas, prodCount)` и `destroyThread(consIdMas, consCount)`, которые удаляют процессы-продюсеры и процессы-потребители.

Добавление и извлечение сообщений: Функции `generateDataMessage(short int size)` и `generateMessage()` добавляют и извлекают сообщения из очереди.

Основной цикл: В функции `main()` находится основной цикл программы, который обрабатывает пользовательский ввод и выполняет соответствующие действия. Пользовательский ввод имеет следующую структуру:

- «1» , создается производитель.
- «2», удаляет последний созданный производитель.
- «3», создается потребитель, который достает сообщения из очереди.
- «4», удаляет последний созданный потребитель.
- «5», вывод информации.
- «q», выход из программы.

Дополнительные функции: Программа также предоставляет функции для вывода информации о процессах, тестирования и вывода инструкций по использованию.

2. ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА.

Проект собирается с помощью makefile. Пример запуска:

```
rlinux@fedora:~$/home/rlinux/Kocherov/ОСиСП/lab04/build/main
```

Для запуска проекта нам требуется в терминале заупустить программу `main`. Где программа сразу переходит в цикл обработки символов

В проекте имеется каталог для сборки debug и release. Каталог git для системы контроля версий моего проекта. Директория src с исходным кодом. И makefile для компиляции и сборки проекта.

3. ПОРЯДОК СБОРКИ И ИСПОЛЬЗОВАНИЯ.

Для компиляции и сборки проекта используется makefile.

Порядок сборки:

1) Задание переменных:

`CC = gcc`: Устанавливает компилятор, который будет использоваться для сборки проекта, в данном случае это GCC.

`CFLAGS = -Wall -Wextra -pthread -lrt -std=c11`: Определяет флаги компиляции, которые будут использоваться, включая включение предупреждений, поддержку многопоточности, время реального времени и стандарт C11.

`SRC_DIR = src`: Указывает директорию, в которой находятся исходные файлы проекта.

`BUILD_DIR = build`: Устанавливает директорию, в которой будут создаваться скомпилированные файлы.

2) Определение компилятора и флагов компиляции:

`CC` - компилятор (gcc).

`CFLAGS` - флаги компиляции, выбираются в зависимости от переменной MODE (отладочная или релизная сборка).

3) Определение зависимостей: `$(BUILD_DIR) /main :$(SRC_DIR) /main.c $(BUILD_DIR)`: Указывает, что цель `$(BUILD_DIR)/main` зависит от файла `$(SRC_DIR)/main.c` и каталога `$(BUILD_DIR)`.

4) Определение целей:

`all` - основная цель, компиляция всех объектных файлов и создание исполняемого файла.

`$(main)`- правило для создания исполняемого файла.

Порядок использования.

1) Компиляция:

Для релизной сборки: `make`

2) Очистка:

`make clean` - удаляет все объектные файлы и исполняемый файл.

3) Запуск:

После успешной компиляции запустите исполняемый файл, например: `./build/main`.

4. МЕТОД ТЕСТИРОВАНИЯ И РЕЗУЛЬТАТ ТЕСТИРОВАНИЯ.

```
rlinux@fedora:~$/home/rlinux/Kocherov/ОСиСП/lab04/build/main
[1] - add producer
[2] - delete producer
[3] - add consumer
[4] - delete consumer
[5]-info
[q]-end
1
[1] - add producer
[2] - delete producer
[3] - add consumer
[4] - delete consumer
[5]-info
[q]-end
MSG ADD: 13952 // 166
EQFc&yIdQ-2M2;:-FLK#ijd2@0,f}C/BqR'wNM[|ZlLk)cwM1BMx/1*L@65=
TK2pnr1/%,Kop3F^_sx;b_p`VP*mDeAlcP\
SCl_Gw,8j>[J{PC85$)tZV}GwdedG7BwY1Y}(b6o#p
PRODUCER COUNT: 1
MSG ADD: 12349 // 155
-m&'Q'g:s|FLPA^LPQKMvy)
(zT1SGZ)RJ.Vz4@5*>XSkx37IaasX[y]WPn,tH4Ho?!lQ>!XYX-FQ>Zw!
<jXvf6O6&Y,KjT=,T)Zo)4J^>m/[I(\dp4\
4
XI,mLdvwPL6Y}<5lDF8"d'1AP6z6(/p_U{L"aCx1l
PRODUCER COUNT: 2
MSG ADD: 834 // 7
A|xVswB1
PRODUCER COUNT: 3
2
[1] - add producer
[2] - delete producer
[3] - add consumer
[4] - delete consumer
[5]-info
[q]-end
3
[1] - add producer
[2] - delete producer
[3] - add consumer
[4] - delete consumer
```

```
[5]-info
[q]-end
BEFORE SEM
AFTER SEM
MSG READ: 13952 // 166
EQFc&yIdQ-2M2;:-FLK#ijd2@0,f}C/BqR'wNM[|ZlLk)cwM1BMx/1*L@65=
VA^HpcA@2y=kghVnMN;\ogV{w_J7r\TK|
2pnr1/%,Kop3F^_sx;b_p`VP*mDeAlcP\
SCl_Gw,8j>[J{PC85$)tZV}GwdedG7BwY1Y}(b6o#p
CONSUMER COUNT: 1
BEFORE SEM
AFTER SEM
MSG READ: 12349 // 155
-m&'Q'g:s|FLPA^LPQKMvy)
(zT1SGZ)RJ.Vz4@5*>XSkx37IaasX[y]WPn,tH4Ho?!lQ>!XYX-FQ>Zw!
<jXvf6O6&Y,KjT=,T)Zo)4J^>m/[I(\dp4\
XI,mLdvwPL6Y}<5lDF8"d'1AP6z6(/p_U{L"aCx1l
CONSUMER COUNT: 2
5
Number of producers: 3
5
Number of consumers: 2
Queue size: 10
Filled slots: 1
Empty slots: 9[1] - add producer
[2] - delete producer
[3] - add consumer
[4] - delete consumer
[5]-info
[q]-end
4
BEFORE SEM
AFTER SEM
MSG READ: 834 // 7
A|xVswB1
CONSUMER COUNT: 3
[1] - add producer
[2] - delete producer
[3] - add consumer
[4] - delete consumer
[5]-info
[q]-end
q
exiting program...
rlinux@fedora:~$
```