ИДЗ №2

Автор работы: Безруков Григорий Александрович, БПИ2310
Ожидаемая оценка: 10

Условие задачи

Задача о курильщиках:

Есть три процесса-курильщика и один процесс-посредник. Курильщик непрерывно скручивает сигареты и курит. Для изготовления сигареты необходимы табак, бумага и спички. У первого курильщика всегда есть табак, у второго — бумага, у третьего — спички. Посредник через случайные промежутки времени выкладывает на стол два любых разных компонента. Курильщик, обладающий третьим компонентом, забирает материалы, скручивает сигарету и курит некоторое время. Новый набор компонентов посредник кладет через фиксированный интервал времени, который может быть больше или меньше времени курения. Если нужный курильщик занят, посредник ждет. Ситуация может развиваться так, что все курильщики будут одновременно курить или наоборот ждать новых материалов.

Необходимо создать многопроцессную модель описанного поведения.

Общее описание решения

Архитектура процессов

Хотя в задаче изначально фигурируют 3 курильщика, разработанное решение позволяет использовать произвольное их количество (обозначим через `n`). Всего в приложении задействовано `n + 4` процессов:

- **Мастер-процесс** — первый процесс, который запускается. Он инициализирует все IPC-объекты и управляет завершением работы;
- **Процесс-производитель** — периодически генерирует новые партии ресурсов и передает их посреднику;
- **Процесс-посредник** — принимает заявки от курильщиков и поставки от производителя, ведёт учёт ресурсов и организует передачу материалов;
- `n` **процессов-курильщиков** — каждый курильщик запрашивает необходимые ресурсы, скручивает сигарету и "курит".

Для удобства запуска процессы-курильщики создаются одним родительским процессом.

Жизненный цикл приложения

Этап 1. Инициализация

- Мастер создает все IPC-объекты (очереди сообщений, разделяемую память, семафоры).
- Рабочие процессы отправляют свои PID'ы мастеру.

- После получения всех PID'ов мастер посылает разрешение на старт каждому процессу.
- Все процессы после получения разрешения переходят к основной фазе работы.

Этап 2. Основная работа

- Курильщики посылают посреднику запросы на ресурсы.
- Производитель периодически отправляет посреднику запросы на пополнение ресурсов.
- Посредник обрабатывает очередь запросов: если на складе есть подходящие ресурсы, посредник выдает их курильщику.
- После скручивания сигареты и "курения" курильщики делают новые запросы.
- Работа продолжается бесконечно до получения сигнала остановки.

Этап 3. Завершение работы

Возможны два сценария завершения:

- **Штатное завершение** — рабочий процесс получает сигнал (`SIGINT`, `SIGTERM`, `SIGQUIT`), уведомляет об этом мастер-процесс и завершает работу. После остановки всех процессов мастер освобождает IPC-объекты.
- **Принудительное завершение** — если сигнал приходит мастеру, он рассылает рабочим процессам `SIGUSR1` для немедленного завершения, сам очищает ресурсы и завершает работу.

Детали реализации

Организация проекта

Проект написан на C++ и имеет следующую структуру:

```
│   choose_out.h
│   CMakeLists.txt
│   constants.h
│   empty.h
│   handshake.h
│   master.cpp
│   merge_outputs.py
│   producer.cpp
│   README.md
│   resources.cpp
│   smokers.cpp
│   SyncCout.h
│
├───name_generator
│       NameGenerator.cpp
│       NameGenerator.h
│
├───pipe
│       base.h
│       get.h
│       posix_mq_based.cpp
```

```
│       semaphore_based.cpp
│       sysv_mq_based.cpp
│
├───semaphore
│       base.h
│       get.h
│       posix_named.cpp
│       posix_unnamed.cpp
│       sysv.cpp
│
├───shared_memory
│       base.h
│       get.h
│       posix.cpp
│       sysv.cpp
│
└───tests
        ...
```

Описание компонентов

- **Основные файлы:**
  - `master.cpp`, `resources.cpp`, `producer.cpp`, `smokers.cpp` — реализация логики всех процессов.
  - Все процессы используют абстрактные интерфейсы для работы с IPC:
    - `sem::Base` — для семафоров (`semaphore/`),
    - `shm::Base` — для разделяемой памяти (`shared_memory/`),
    - `pipes::Base` — для передачи сообщений (`pipe/`).
- **Реализации интерфейсов:**
  - Семафоры: POSIX именованные, POSIX неименованные, System V.
  - Разделяемая память: POSIX или System V.
  - Каналы сообщений: через очередь сообщений POSIX, очередь сообщений System V или через семафоры + память.
- **Дополнительные файлы:**
  - `choose_out.h` — обработка аргументов командной строки.
  - `constants.h` — основные настройки (например, количество курильщиков).
  - `handshake.h` — процедуры рукопожатия между мастером и рабочими процессами.
  - `SyncCout.h` — поток вывода с синхронизацией для красивого логирования.
  - `merge_outputs.py` — скрипт для объединения логов процессов по времени.
  - `name_generator/NameGenerator.(h|cpp)` — генератор имен IPC-объектов.

Инструкции по запуску

- Каждую программу (`master`, `producer`, `resources`, `smokers`) нужно запускать отдельно.

- Первый запускается **обязательно** мастер.
- Далее остальные процессы можно запускать в произвольном порядке.
- Можно передать путь к файлу для вывода логов как аргумент командной строки. Иначе лог пишется в стандартный поток вывода.

Тестирование

Для каждой версии решения проведено тестирование. Логи работы процессов сохранены в папке `tests/`.
Чтобы упростить проверку, все выходные данные были объединены с помощью скрипта `merge_outputs.py`, сортирующего записи по времени создания.

Отмечу, что из-за высокой скорости некоторых операций метки времени могут совпадать, поэтому возможны небольшие отличия в порядке событий.