**Testy**

**Test 2.1**

W celu zbadania opóźnienia występującego między momentem udostępnienia danych przez producenta a odebraniem ich przez klienta przy różnych metodach wymiany informacji, użyliśmy:

* Pamięci współdzielonej, synchronizowanej przy użyciu semaforów.
* Wstawiania do bazy danych przy użyciu transakcji.

**Pamięci współdzielonej, synchronizowanej przy użyciu semaforów.**

Program producencki wstawiał treść wiadomości do pliku, z którego następnie były one pobierane przez program kliencki zajmujący się cenzurowaniem. Dostęp do pliku zabezpieczony był poprzez semafory.

**Wstawiania do bazy danych przy użyciu transakcji.**

Program producencki po wygenerowaniu wiadomości wstawiał ją do bazy danych. Do bazy wstawiane były treści wiadomości i czas wstawienia. Program kliencki pobierał te dane, po czym usuwał je z tej bazy. Dostęp do bazy odbywał się za pomocą transakcji.

**Archiwizacja**

Po wysłaniu wiadomości do pliku/bazy danych producent wysyłał dane o wiadomości do bazy archiwizującej cały proces. W bazie tej przez producenta umieszczone zostały: treść wygenerowanej wiadomości, czas i data wysłania wiadomości, oraz czas wysłania wiadomości w mikrosekundach. Klient po ocenzurowaniu wiadomości wysyłał dane o swoich działaniach, do tej samej bazy danych. Są to treść wiadomości po dokonaniu cenzury, czas odbioru wiadomości - czas wczytania z pliku/ bazy danych i różnica między czasem odebrania i wysłania wiadomości. Informacje dotyczące jednej wiadomości przetrzymywane są w jednym rekordzie bazy. Obie tabele dostępne w repozytorium.

**Test\_2\_1\_sem.cvs** - z użyciem pliku i semaforów.

**Test\_2\_1\_db.cvs** - z użyciem bazy i transakcji.

**archive.db -** wszytstkiewyniki testów.

**Wyniki:**

|  |  |  |
| --- | --- | --- |
| rodzaj synchronizacji | plik i semafor | baza danych i transakcje |
| liczba wiadomości | 10000 | 10000 |
| średni odstęp | 19.8405 | 75.6461646164616 |

**Test 2.2**

System został dodatkowo sprawdzony pod kątem użycia różnych polityk szeregowania. Sprawdziliśmy, czy zmiana polityki szeregowania wpłynie i w jaki sposób, na regularność dostarczania danych przez program producencki. Zgodnie z informacjami pozyskanymi na wykładzie, zastosowaliśmy trzy różne algorytmy szeregowania. SCHED\_OTHER, SCHED\_BATCH, oraz SCHED\_FIFO. Użyliśmy do tego funkcji sched\_setscheduler. SCHED\_OTHER jest typowym domyślnym trybem, gdzie zadania po kolei dostają kwant czasu. Dla tego trybu przeprowadziliśmy pierwsze eksperymenty. Następnie przeszliśmy do SCHED\_BATCH, trybu, który zapewnia większą przepustowość przy możliwości wystąpienia większych opóźnień. Jako ostatni został przetestowany SCHED\_FIFO. SCHED\_FIFO działa jak kolejka FIFO, to znaczy, że pierwszy zgłoszony proces był obsługiwany w pierwszej kolejności, bez jakichkolwiek ograniczeń czasowych. W przypadku wywłaszczenia procesu w tym trybie, przez proces o większym priorytecie, po zakończeniu procesu wywłaszczającego, zostaje wznowiony proces wywłaszczony. Procesy w tym trybie są wykonywane do momentu zakończenia się, zawieszenia się na operacji I/O, bądź też wywołania funkcji sched\_yield.

|  |  |  |
| --- | --- | --- |
|  | średnia | odchylenie |
| SCHED\_FIFO | 125821,6571 | 5791,19296 |
| SCHED\_BATCH | 301900,4615 | 261326,095 |