Marcin Cieślikowski

Marek Jasiński

Paweł Goździkowski

**Dokumentacja końcowa – projekt UXP1A**

**Treść**: Napisać program DLM (Distributed Lock Manager) - zarządcy blokad. Celem DLM jest synchronizacja dostępu do współdzielonych zasobów. Aby bezpiecznie korzystać z zasobu należy założyć na niego blokadę. Jeżeli założenie blokady nie jest możliwe proces żądający blokady musi być zawieszony do czasu aż wystąpi timeout lub blokada będzie mogła być przez niego uzyskana.

**1) Interpretacja treści zadania**

Należy napisać program zarządcy blokad oraz bibliotekę funkcji, dzięki którym procesy działające na współdzielonych zasobach, będą synchronizowane w dostępie do tych zasobów. Komunikacja i synchronizacja pomiędzy zarządcą blokad, a pozostałymi procesami będzie się odbywać poprzez potoki nazwane (FIFO). Procesy w celu uzyskania dostępu do zasobu będą musiały wywołać funkcję biblioteczne. Procesy mogą zakładać różne typy blokad tj. Concurrent Read, Concurrent Write, Protected Read, Protected Write oraz Exclusive. Proces zarządcy blokad rozstrzygał będzie o możliwości założenia blokady przez proces na podstawie aktualnie założonych blokad, zgodnie z macierzą współzależności między typami blokad. W celu uniknięcia zagłodzenia możliwość przydzielenia procesowi zasobu jest rozpatrywana wyłącznie wtedy, gdy inny proces, który wcześniej zgłosił żądanie, nie czeka na ten zasób.

Zasoby są przydzielane wg. poniżej zdefiniowanej macierzy:

Rys. Macierz koincydencji

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | CR | CW | PR | PW | EX |
| CR | + | + | + | + | - |
| CW | + | + | - | - | - |
| PR | + | - | + | - | - |
| PW | + | - | - | - | - |
| EX | - | - | - | - | - |

**2) Pełen opis funkcjonalny – „black-box”**

Proces DLM – przy starcie inicjalizuje potok nazwany (FIFO), który służyć będzie do odbierania komunikatów od procesów. Potok ma stałą i znaną nazwę, oraz ścieżkę dostępu. Jest to /tmp/DLM/DLMfifo. Proces inicjalizuje struktury przechowujące informację o zasobach. Proces zawiesza się w oczekiwaniu na żądania dostępu lub zwolnienia zasobów współdzielonych. Żądania obsługiwane są w kolejności zgłoszeń.

**3) Pełen opis biblioteki klienckiej:**

**DLM\_PATH - ścieżka do katalogu w którym będą przechowywane pliki potoków nazwanych (FIFO).**

**DLM\_FIFO\_PATH - ścieżka do kolejki DLM.**

**Typy blokad :**

|  |  |  |
| --- | --- | --- |
| Pełna nazwa  blokady | Zdefiniowana  nazwa | Kod blokady |
| Concurrent Read | CR | 0 |
| Concurrent Write | CW | 1 |
| Protected Read | PR | 2 |
| Protected Write | PW | 3 |
| Exclusive | EX | 4 |

**Kody odpowiedzi:**

**GRANTED** - przydzielono zasób.

**TIMEDOUT** - wystąpił timeout - nie przydzielono zasobu.

**LOCKED** - zasób zajęty.

**UNLOCKED** - zasób został zwolniony.

**FREE** - zasób jest wolny, zwracany w obsłudze żądania TRYLOCK, która nie przydziela zasobu, a tylko sprawdza, czy zasób jest wolny.

**Kody błędów:**

**EOPENDLMFIFO** - błąd otwarcia potoku nazwanego (FIFO) DLM-a.

**ECREATEFIFO** - bład utworzenia potoku nazwanego - klienta, bądź DLM-a (ten błąd może wynikać z nieposiadania uprawnień do katalogu podanym w DLM\_PATH).

**EOPENCLIENTFIFO** - błąd otwarcia potoku nazwanego klienta.

**EWRITE** - błąd wysyłania żądania do DLM-a, występuje wtedy, gdy nie uda się wysłać pełnej struktury DLMrequest.

**EREAD** - błąd czytania z potoku - występuje wtedy, gdy nie uda nam się odczytać pełnej stuktury DLMresponse.

**EBADLOCKTYPE** - niepoprawna blokada - podana inna blokada niż te zdefiniowane w punkcie o typach blokad.

**EBADTIMEOUT** - niepoprawny timeout - podana wartość ujemna < -2.

**ENOTLOCKED** - próba zwolnienia zasobu, który nie został przydzielony.

**EAGAIN** - próba ponownego zajęcia tego samego zasobu.

**- int DLM\_lock(int resource\_id, int lock\_type, long timeout)** – wysyła komunikat do zarządcy blokad. Komunikat składa się z identyfikatora zasobu, typu blokady, czasu timeout, identyfikatora procesu (PID). Następnie tworzony jest potok nazwany (FIFO) o nazwie odpowiadającej pid procesu. Proces zawiesza się w oczekiwaniu na komunikat od zarządcy blokad. Typy komunikatów zwrotnych od procesu zarządcy:

Po odebraniu komunikatu proces usuwa potok.

**- int DLM\_unlock(int resource\_id)** – wysyła do DLM żądanie zwolnienia zasobu składające się z identyfikatora zasobu oraz identyfikatora procesu (PID). Jako typ blokady przekazywany jest specjalny parametr FREERESOURCE. Parametr timeout nie ma tutaj znaczenia. Następnie tworzony jest potok nazwany (FIFO) o nazwie odpowiadającej pid procesu. Proces zawiesza się w oczekiwaniu na komunikat od zarządcy blokad. Typy komunikatów zwrotnych od procesu zarządcy:

**- int DLM\_trylock(int resource\_id, int lock\_type)** – wysyła żądanie, w którym parametr timeout = TRYLOCK, gdzie TRYLOCK oznacza zdefiniowaną stałą. DLM odpowiada na takie zgłoszenie możliwie najszybciej. Typy komunikatów zwrotnych procesu zarządcy:

Po odebraniu komunikatu proces usuwa potok.

**4) Opis i analiza poprawności stosowanych protokołów komunikacyjnych**

Rysunki przedstawiają strukturę komunikatów przesyłanych do i od DLM. Komunikaty obsługiwane są przez DLM w kolejności zgłaszania, co wynika ze struktury potoku FIFO. Proces oczekujący na odpowiedź od DLM zawiesza się na utworzonym potoku w oczekiwaniu na tę odpowiedź. Po odebraniu wiadomości potok odbiorczy procesu jest kasowany, dzięki czemu zakończone procesy nie powodują pozostania nieużywanych potoków. Nazwa potoku odbiorczego procesu jest ściśle ustalona i zawiera numer identyfikacyjny procesu (PID). Aby komunikaty były przesyłane atomowo muszą być mniejsze niż pojemność bufora kolejki FIFO. Pojemność bufora kolejki w Linuxie to 4096B, a standard POSIX 1-2001 wymaga bufora większego od 512B. Komunikaty do i z DLM mają odpowiednio 18B i 4B, dzięki czemu będą przesyłane atomowo.

Struktura opisująca komunikat pokazany na Rys. 2:

typedef struct DLMrequest {

pid\_t pid;

int resource\_id;

int lock\_type;

long timeout;

} DLMrequest;

Struktura opisująca komunikat pokazany na Rys. 3 :

typedef struct DLMresponse {

int response;

} DLMresponse;



Rys 2. Komunikat do DLM.



Rys 3. Komunikat odpowiedzi DLM.

**4) Podział na moduły i struktura komunikacji między nimi**

Rysunek przedstawia schemat podziału na moduły oraz strukturę komunikacji pomiędzy nimi.



**5) Opis wykorzystanych narzędzi:**

- biblioteka kliencka DLM jest napisana w języku C i korzysta z podstawowych bibliotek języka C.

- DLM napisany jest w języku C++ (użyte zostały kontenery STL).

- mechanizmy systemu unix/linux (sygnały, kolejki nazwane(FIFO)).

**6) Uruchamianie:**

- proces DLM można uruchomić z następującymi flagami:

-d - uruchamia w trybie demona.

-l <nazwa\_pliku> - logowanie do pliku.

-h wyswietla pomoc programu.

**Opcja demona musi być użyta w parze z opcją logowania do pliku.**