Maciej Dmowski – Grupa pn. 10:15-12:00 CS500

Systemy operacyjne - Laboratorium 3

Koncepcja rozwiązania

## 1. Treść zadania

1. Zaimplementować mechanizm priorytetowych kolejek wiadomości dla komunikacji międzyprocesowej.
2. Wykorzystując zaimplementowany mechanizm kolejkowy przygotować części składowe systemu przetwarzającego wiadomości w kolejkach.
3. Przeprowadzić prezentację systemu, odpowiedzieć na pytania.

## 2. Kolejki priorytetowe

Rozwiązanie zawierać będzie 3 kolejki, z zapisanym wskaźnikiem na początek i koniec każdej kolejki, w celu łatwiejszego dodawania i pobierania elementów oraz zmienną *capacity* inicjalizowaną określoną wartością określającą pojemność kolejki oraz zmienną *size* informującą o obecnym rozmiarze kolejki.

Kolejki zostaną zaimplementowane jako listy jednokierunkowe, których elementy będą alokowane w pamięci współdzielonej i łączone ze sobą zgodnie ze schematem listy jednokierunkowej. Dodanie elementu do pamięci współdzielonej zaimplementowane zostanie przez funkcję *mmap*. Elementy jako struktura zawierać będą:

* Komunikat
* Priorytet
* stempel czasowy ustawiany przy dodawaniu elementu do kolejki.

Dodanie elementu o priorytecie zwykłym dodawać będzie element na koniec kolejki a elementu priorytetowego na początek kolejki jeśli nie ma jeszcze takiego elementu, a gdy jest to za ostatnim istniejącym elementem priorytetowym. Dodatkowo zostanie zapewniona funkcja zwracająca obecny stopień zapełnienia kolejki w postaci podania zmiennej *size* oraz *capacity*.

## 3. System przetwarzający wiadomości w kolejkach

Każda z kolejek będzie obsługiwana przez 3 semafory:

* 2 zliczające
  + *empty* - Liczba pustych miejsc – inicjalizowane pojemnością kolejki
  + *full* - Liczba zajętych miejsc – inicjalizowane wartością 0
* 1 mutex
  + *mutex* - Realizujący wzajemne wykluczanie procesów chcących otrzymać dostęp do kolejki

Producent sprawdza czy jest miejsce na kolejny element (sprawdzenie wartości semafora *empty*), jeśli jest sprawdza czy ma dostęp do kolejki (semafor *mutex)* i umieszcza nowy element w odpowiednim miejscu – w zależności od priorytetu, następnie podnosi semafor *full* i podnosi semafor *mutex*. W przypadku braku wolnych miejsc zwraca informację o przepełnieniu w celach testowych oraz oczekuje na miejsce.

Producent specjalny będzie korzystał z elementów kodu producenta jedynie kolejka, do której będzie produkował wiadomości będzie losowana.

Konsument sprawdza czy kolejka nie jest pusta poprzez sprawdzenie wartości semafora *full*, jeśli jest większy od 0 to sprawdza czy ma dostęp do kolejki poprzez sprawdzenie semafora *mutex* i pobiera element i upuszcza semafor *full* oraz podnosi *mutex.* Następnie zgodnie z danym prawdopodobieństwem pr dopisuje lub nie nową literę, po sprawdzeniu symbolu sprawdza czy ma dostęp do danej kolejki poprzez jej semafor *mutex*, po czym zapisuje ten element w odpowiednim miejscu w zależności od priorytetu.

Oczekiwanie między działaniami realizowane będzie poprzez funkcję *sleep()*.

Każdy z typów procesów opisany zostanie przez funkcje wykonujące opisane w poleceniu zadania i korzystający z mechanizmów synchronizacji – semaforów. Pozwoli to na wielokrotne użycie kodu i generowanie producentów oraz konsumentów przekazując im jako argument wskaźnik na „głowę” danej kolejki.

1. Prezentacja systemu

System zostanie zaprezentowany poprzez stworzenie trzech procesów producentów, trzech konsumentów oraz jednego producenta specjalnego. W celu weryfikacji założeń każda kolejka będzie wypisywać swój stan do oddzielnego pliku tekstowego oraz czas obsługi kolejnych komunikatów. Cały proces przerywany będzie automatycznie po kilku minutach i wypisywane będą dodatkowo średnie czasy obsługi komunikatu ze względu na priorytet. Testy zostaną wykonane dla kilku wartości pr oraz pojemności kolejek. Odpowiedź na pytania zostanie poparta wynikami programu testowego.