Nieustraszeni zabójcy chomików – algorytm

Anna Lubawa 136762, Michał Cichy 132203

Struktury

1. **Resource** – zawiera informacje o danym rodzaju zasobu, o który będą rywalizować procesy.

    1.1. typ zasobu (zlecenia/agrafki/trucizna)

    1.2. całkowitą liczbę dostępnych jednostek zasobu

    1.3. listę zgłoszeń o zasób (lista Requestów)

    1.4. listę odpowiedzi od procesów (mają funkcję synchronizacyjną, =ACK z algorytmu Lamporta)

2. **Request –** opisuje strukturę wiadomości dotyczącej żądania zasobu, zawiera:

    2.1. id procesu wysyłającego

    2.2. liczbę jednostek żądanego zasobu

    2.3 znacznik czasowy (wg zegara Lamporta)

3. **Release** - struktura wysyłana do innych procesów informująca o zwolnieniu zasobu

    3.1. id procesu wysyłającego

    3.2. liczbę jednostek zwalnianego zasobu

    3.3 znacznik czasowy (wg zegara Lamporta)

4. **Reply** – opisuje strukturę odpowiedzi procesu na żądanie zasobu przez inny proces, służy do synchronizacji, zawiera:

    4.1. id procesu wysyłającego

    4.2. znacznik czasowy (wg zegara Lamporta)

5. **Zlecenie** – wygenerowane przez „burmistrza” zlecenie, zawiera:

    5.1 liczbę chomików do zabicia w ramach tego zlecenia

Typy wiadomości

INIT_ZLECENIA – wiadomość wysyłana przez „burmistrza” na początku działania programu do wszystkich procesów zawierająca listę zleceń

ZLECENIE_REQUEST – wiadomość wysyłana przez proces do wszystkich innych (oprócz „burmistrza”), oznacza żądanie otrzymania zlecenia i zawiera strukturę typu Request

ZLECENIE_REPLY - wiadomość wysyłana przez wszystkie procesy (oprócz „burmistrza”) do procesu który wysłał wiadomość *ZLECENIE_REQUEST*

AGRAFKI_REQUEST - wiadomość wysyłana przez proces do wszystkich innych (oprócz „burmistrza”), oznacza żądanie otrzymania agrafki i zawiera strukturę typu Request

AGRAFKI_REPLY - wiadomość wysyłana przez wszystkie procesy (oprócz „burmistrza”) do procesu który wysłał wiadomość *AGRAFKI_REQUEST*

AGRAFKI_RELEASE - wiadomość wysyłana przez proces do wszystkich innych (oprócz „burmistrza”), oznacza zwolnienie danej liczby agrafek; agrafki zwalniamy, a zleceń nie, bo zlecenia są jednorazowe, więc nie potrzeba ich zwalniać.

TRUCIZNA_REQUEST - wiadomość wysyłana przez proces do wszystkich innych (oprócz „burmistrza”), oznacza żądanie otrzymania odpowiedniej porcji trucizny, zależnej od wielkości zlecenia; wiadomość zawiera strukturę typu Request

TRUCIZNA_REPLY - wiadomość wysyłana przez wszystkie procesy (oprócz „burmistrza”) do procesu który wysłał wiadomość *TRUCIZNA_REQUEST*

TRUCIZNA_RELEASE - wiadomość wysyłana przez proces do wszystkich innych (oprócz „burmistrza”)

ZLECENIE_COMPLETED – wiadomość wysyłana do „burmistrza” przez proces, który wykonał swoje zlecenie

STOP – wiadomość wysyłana przez „burmistrza” do pozostałych procesów sygnalizująca wykonanie danej partii zleceń

Stany procesów

INITIAL – stan początkowy, procesy odbierają wiadomość typu *INIT_ZLECENIA*, pozostałe wiadomości ignorują.

BURMISTRZ_WAITING – w tym stanie znajdować się może jedynie proces burmistrza, następuje on po *INITIAL* – proces odbiera w nim jedynie wiadomości typu *ZLECENIE_COMPLETED* od skrzatów, dzięki temu wie kiedy wszystkie zlecenia zostały wykonane. Kiedy to nastąpi wysyła do wszystkich skrzatów wiadomość typu *STOP* sygnalizującą żeby procesy przeszły do stanu *INITIAL*, następnie również przechodzi do tego stanu.

SKRZAT_REQUESTING – stan ubiegania się o zasoby, dotyczy tylko procesów-skrzatów. Odbierane są wszystkie wiadomości oprócz *INIT_ZLECENIA*, które jest ignorowane i *ZLECENIE_COMPLETED*, które może być wysyłane tylko do burmistrza, więc proces-skrzat go nigdy nie otrzyma. Po uzyskaniu odpowiedzi REPLY od wszystkich skrzatów skrzat przechodzi w stan SKRZAT_CRITICAL.

Reakcje na wiadomości w tymże stanie:

STOP – proces przechodzi do stanu *INITIAL*

REPLY - jeżeli odpowiedź jest odpowiedzią nr n - 1, skrzat przechodzi do stanu *SKRZAT_CRITICAL*.

SKRZAT_CRITICAL - skrzat uzyskał odpowiedź *REPLY* od innych skrzatów i przegląda listę requestów posortowanych chronologicznie w celu ustalenia, czy starczy dla niego zasobu. Jeśli starczy, to przechodzi do *SKRZAT_REQUESTING* w związku z kolejnym zasobem. Jeśli nie starczy, to przechodzi w stan *SKRZAT_WAIT_FOR_RELEASE.*

SKRZAT_WAIT_FOR_RELEASE - skrzat pozostaje bezczynny aż do nadejścia wiadomości *RELEASE*. Wtedy z powrotem przechodzi w stan *SKRZAT_REQUESTING*.

SKRZAT_FINISHED – proces-skrzat skończył wykonywać zlecenie – zwalnia zasoby (agrafki, trucizny) i wysyła burmistrzowi wiadomość typu *ZLECENIE_COMPLETED*, a następnie ponownie przechodzi do stanu *SKRZAT_REQUESTING*, żeby ubiegać się o ewentualne dostępne jeszcze dodatkowe zlecenia. W razie wiadomości *STOP* od burmistrza wróci wtedy do stanu *INIT* i będzie czekał na kolejne rozdanie wiadomości od burmistrza.

Opis algorytmu

1. Liczba dostępnych agrafek i trucizn oraz liczba możliwych zleceń generowanych przez proces burmistrza jest określana przy uruchomieniu programu.

2. „Burmistrz”, czyli główny proces, generuje listę zleceń – każde zlecenie to pewna liczba całkowita określająca ile chomików ma być zabitych. Lista wysyłana jest do wszystkich procesów – wiadomość typu *INIT_ZLECENIA*.

Burmistrz przechodzi ze stanu *INITIAL* w stan *BURMISTRZ_WAITING*.

3. Procesy po otrzymaniu listy zleceń inicjują swoje struktury przechowujące informacje o każdym z zasobów (struktura Resource, osobna instancja dla każdego z zasobów). Inicjalizacja polega na zapisaniu liczby maksymalnie dostępnych zasobów tj. liczby wszystkich zleceń, liczby dostępnych agrafek oraz liczby dostępnych porcji trucizny (są to 3 osobne struktury).

4. Każdy z procesów „skrzatów” uruchamia wątki nasłuchujące: osobny na żądania, osobny na odpowiedzi i osobny na zwolnienia zasobów przez inne procesy. Uruchamia też dodatkowy wątek do nasłuchiwania na wiadomość typu *STOP* od burmistrza.

Fundamentem algorytmu jest utrzymywanie dla każdego procesu, dla każdego zasobu osobnej **listy żądań** typu Request (wewnątrz struktury Resource), zawierającej żądania innych procesów ze znacznikami czasowymi z zegarów Lamporta.

Wszystkie skrzaty, które ubiegają się o zasób i dostały REPLY (czyli ACK) od wszystkich pozostałych skrzatów, przeglądają swoje lokalne listy żądań posortowanych wg znaczników czasowych. Sumują liczby jednostek z kolejnych żądań i jeśli po dotarciu do swojego własnego żądania na tej liście maksymalna ilość zasobu pomniejszona o naliczoną sumę jest równa lub większa ilości żądanej przez danego skrzata, skrzat ten otrzymuje dany zasób.

Po wykorzystaniu zasobu skrzat wysyła wiadomość typu RELEASE do pozostałych. Wtedy wątki nasłuchujące na wiadomości typu Release usuwają odpowiednie żądania z lokalnych list żądań, dzięki czemu skrzaty w stanie *WAITING* będą miały szansę na zajęcie zasobu.

Podobny schemat aplikuje się do wszystkich rodzajów zasobów. W przypadku zleceń procesy zawsze żądają jednej jednostki i zlecenia są przypisywane - nr zlecenia równy pozycji żądania procesu na liście żądań. **Jeżeli zleceń jest więcej niż skrzatów**, to po wykonaniu jednego zlecenia skrzaty **mają szanse na wykonanie kolejnego** jeszcze przed “nowym rozdaniem” pochodzącym od burmistrza. Na liście żądań pojawią się po prostu nowe, których indeksy będą odpowiadały jeszcze nie przydzielonym zadaniom. **W przeciwieństwie do innych zasobów żądania dotyczące zleceń nie są usuwane po wykonaniu, bo zlecenia nie są odzyskiwalne**. W takiej sytuacji nie będzie problemu z indeksami i przypisaniem zadania do żądania.

5. Skrzaty przechodzą ze stanu *INITIAL* w stan *SKRZAT_REQUESTING*. Procesy „skrzaty” zaczynają ubiegać się o zlecenia – wysyłają wiadomość typu *ZLECENIE_REQUEST*, a następnie czekają na odpowiedzi od pozostałych procesów (wiadomość *ZLECENIE_REPLY*).

6. Po otrzymaniu wszystkich odpowiedzi skrzat przechodzi w stan *SKRZAT_CRITICAL*: przegląda posortowane chronologicznie (według znaczników czasowych) żądania zlecenia wszystkich procesów i sprawdza, czy zostało dla niego zlecenie. Jeśli tak, przechodzi do ubiegania się o agrafki - stan *SKRZAT_REQUESTING*. Jeśli nie, przechodzi do stanu *INITIAL* i czeka na nową partię zleceń.

7. Jeśli udało się otrzymać zlecenie, proces wysyła wiadomość typu *AGRAFKI_REQUEST*, a następnie czeka na odpowiedzi od pozostałych procesów (wiadomość *AGRAFKI_REPLY*).

8. Po otrzymaniu wszystkich odpowiedzi (przechodzi do stanu *SKRZAT_CRITICAL*) proces przegląda posortowane chronologicznie (według znaczników czasowych) żądania agrafek wszystkich procesów i sprawdza, czy starczy dla niego agrafek. Jeśli tak, przechodzi do ubiegania się o truciznę (*SKRZAT_REQUESTING*). Jeśli nie, ponownie przechodzi do stanu *SKRZAT_WAIT_FOR_RELEASE*.

9. Jeśli udało się otrzymać agrafkę, proces wysyła wiadomość typu *TRUCIZNA_REQUEST*, a następnie czeka na odpowiedzi od pozostałych procesów (wiadomość *TRUCIZNA_REPLY*).

10. Po otrzymaniu wszystkich odpowiedzi (przechodzi do *SKRZAT_CRITICAL*), proces przegląda posortowane chronologicznie (według znaczników czasowych) żądania porcji trucizn wszystkich procesów i sprawdza, czy starczy dla niego trucizny. Jeśli tak - może zakończyć wykonywanie zlecenia (stan *SKRZAT_FINISHED*). Jeśli nie, przechodzi do stanu *SKRZAT_WAIT_FOR_RELEASE*.

11. Po wykonaniu zlecenia proces zwalnia zasoby wysyłając kolejno wiadomości typu *AGRAFKI_REL* i *TRUCIZNA_REL* do pozostałych skrzatów, a następnie do burmistrza wiadomość typu *ZLECENIE_COMPLETED*.