Przemysław Szymoniak

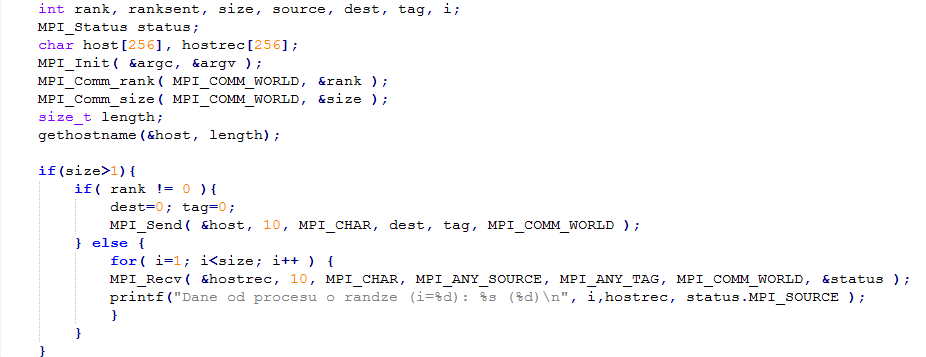
Przetwarzanie współbieżne. Programowanie równoległe i rozproszone.

Sprawozdanie z laboratorium 12.

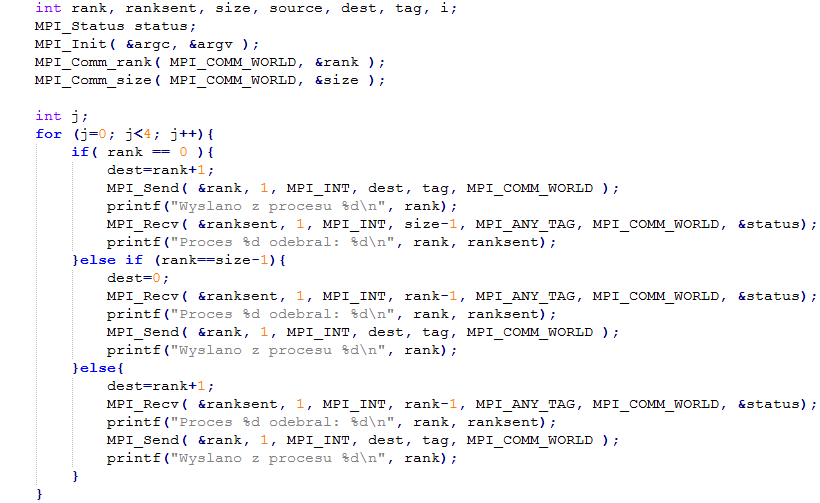
Celem laboratorium opanowanie podstaw programowania z przesyłaniem komunikatów MPI.

W ramach zajęć zrealizowałem następujące kroki:

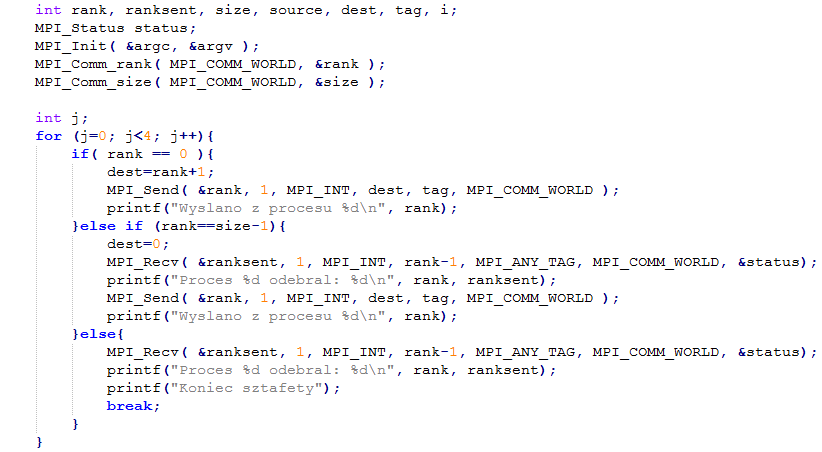
* Pobrałem i rozpakowałem wszystkie wymagane pliki, utworzyłem wymagane katalogi oraz uruchomiłem program.
* Uzupełniłem program o przesyłanie w tablicy znaków adresu internetowego węzła nadawcy:



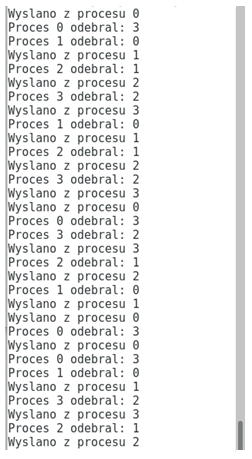
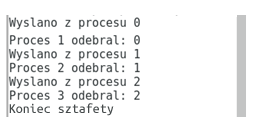
* Kod sztafety (zamknięty pierścień):



* Sztafeta (ostatni kończy):



* Działanie:



Wnioski:

* Polecenie MPI\_send(…) jest blokującą procedurą wysyłania w MPI. Blokująca oznacza, że sterowanie jest przekazywane z powrotem po wykonaniu pełnego wysłania danych, nie tak jak w przypadku procedury nie blokującej MPI\_Isend(…), gdzie wysyłanie jest jedynie inicjowane a następnie sterowanie wraca. Wysłanie danych odbywa się później np. przy użyciu innego wątku.
* Polecenie MPI\_recv(…) jest tak samo jak poprzednik procedurą blokującą, próba odebrania danych więc odbywana się od razu po wywołaniu funkcji a sterowanie wraca w momencie uzyskania danych. Procedurą nie blokującą jest MPI\_Irecv(…), która również tylko inicjuje odbieranie danych odbywające się z opóźnieniem.
* Polecenia MPI\_send(…) oraz MPI\_recv(…) posiadając możliwość określenia do którego wątku dane mają zostać wysłane oraz od którego odebrane. Dodatkowo funckja MPI\_recv(…) posiada argument MPI\_Status zwracający informację o statusie wykonania operacji.
* Wykorzystując MPI jesteśmy w stanie wysyłać różne typy danych zaczynając od CHAR kończąc na BYTE. Większość nazw typów pokrywa się z tymi wykorzystywanymi w programowaniu wymagają one jednak wykorzystania przedrostka MPI\_ np.(MPI\_DOUBLE).
* W naszym algorytmie wykorzystujemy również funkcje MPI\_Comm\_rank(…) jest ona odpowiedzialna za pobranie numeru, identyfikatora wywołującego procesu zapisuje go w zmiennej rank typu int \*.
* Natomiast funkcja MPI\_Comm\_size(…) odpowiada za pobranie liczby procesów występujących w obrębie komunikatora MPI\_Comm zapisując tą wielkość do zmiennej typu int \* o nazwie size;
* Funkcjami również wymaganymi do działania aplikacji jest MPI\_Init(…) odpowiedzialna za inicjowanie środowiska wykonywania programu. Jest to między innymi tworzenie domyślnego komunikatora MPI\_COMM\_WORLD.
* Na zakończenie wykorzystywania MPI powinniśmy zwolnić wszystkie używane zasoby. Możemy to wykonać funkcją MPI\_Finalize(). Dodatkowo ta funkcja przygotowuje system do zamknięcia.
* Odbierając informacje od wątków możemy podać konkretny identyfikator. Jednak w momencie gdy nie wiemy z którego wątku przyjdą oczekiwane informacje możemy wykorzystać flagę MPI\_ANY\_SOURCE. Jej ustawienie sprawia, że wątek odczyta pierwszy natrafiony komunikat od dowolnego procesu.
* Ustawienie MPI\_ANY\_TAG powoduje, że podczas odbierania nie będziemy sprawdzać znacznika typu wiadomości jedynie odbierzemy komunikat, który przyjdzie najwcześniej.