ZADANIE NR 2

W systemie dostępne są (wirtualne) "zasoby". Istnieje *K* różnych rodzajów zasobów (*1 <= K <= 99*) numerowanych od *1* do *K*. Początkowo dostępnych jest *N* sztuk zasobów każdego rodzaju (*2 <= N <= 9999*).

Z zasobów tych do wykonania (równie wirtualnych) prac korzystają procesy `klient`, przy czym wykonują one prace parami.

Dobieraniem klientów w pary oraz kontrolą dostępności zasobów zajmuje się wielowątkowy `serwer`. W zadaniu należy zaimplementować oba te programy.

Serwer

Serwer uruchamia się poleceniem

> ./serwer *K N*

przekazując wartość parametrów opisanych powyżej.

Serwer inicjuje swój stan — początkowo zasoby wszystkich rodzajów są dostępne w liczbie *N* sztuk, tworzy kolejki potrzebne do komunikacji i oczekuje na zgłoszenia klientów.

Serwer dobiera klientów w pary. Jeśli jakiś klient wysyła zgłoszenie o *n* sztuk zasobu *k*, to serwer jako jego parę dobiera kolejnego klienta żądającego zasobów tego samego rodzaju *k* (niech będzie to żądanie *m* sztuk). Dopiero wtedy wątek serwera

- Przydziela *m+n* zasobów tej parze (być może czekając na dostępność zasobów).
- Wypisuje komunikat
  `Wątek id_wątku przydziela m+n zasobów k klientom PID1 PID2, pozostało i zasobów`
  (za zmienne pisane *kursywą* należy wstawić wartości liczbowe, *i* oznacza liczbę zasobów pozostałych po przydzieleniu).
- Powiadamia klientów, wysyłając także każdemu klientowi PID jego partnera.

Obsługą klientów mają zajmować się wątki tworzone i usuwane dynamicznie przez serwer, dla każdego klienta jeden wątek lub ewentualnie jeden wątek na parę klientów. Jedynie pierwsze komunikaty przysyłane przez poszczególnych klientów mogą być odbierane przez jeden wspólny wątek (może to być, ale nie musi, wątek główny serwera).

Jednocześnie może trwać wiele prac (dla różnych jak i dla tych samych rodzajów zasobów — ograniczeniem jest tylko liczba zasobów) oraz może być kompletowanych wiele par (dla różnych rodzajów zasobów). Zasoby stają się dostępne dopiero po zakończeniu pracy przez obu klientów z pary.

Praca serwera może zostać przerwana sygnałem SIGINT (Ctrl+C). W takiej sytuacji serwer powinien zwolnić zajmowane zasoby systemowe (w szczególności kolejki IPC) i zakończyć się. Klienci także mają się w takiej sytuacji zakończyć, mogą w sposób "wyjątkowy".

W zależności od implementacji dopuszczalne jest oczekiwanie na zwolnienie wszystkich przydzielonych już zasobów przez klientów; serwer nie powinien natomiast przyjmować już nowych zleceń.

Serwer powinien działać potencjalnie dowolnie długo i być w stanie obsłużyć nieograniczenie wielu klientów w czasie swojego życia.

Synchronizacja

Do realizacji wielowątkowości na serwerze należy użyć biblioteki pthreads.

Synchronizację między wątkami należy zrealizować wyłącznie za pomocą mechanizmu mutexów i zmiennych warunkowych.

Należy zapewnić żywotność rozwiązania, a w szczególności nie zagłodzić par oczekujących na większą liczbę zasobów. Dla potrzeb oceny poprawności rozwiązania można przyjąć, że klienci otrzymujący zasoby na pewno je zwrócą, a także że mechanizmy pthreads są żywotne.

Klient

Klienta uruchamia się poleceniem

./klient *k n s*

gdzie parametry oznaczają:

- *k* – rodzaj zasobu (*1 <= k <= K*)
- *n* – żądana liczba sztuk zasobu (*1 <= n <= podłoga(N/2)*)
- *s* – czas na jaki zatrzyma się klient przetrzymując zasób (w sekundach). *s* może być równe zero!

Klient wysyła na serwer żądanie *n* sztuk zasobu *k*. Gdy serwer dobierze mu parę i zarezerwuje zasoby, klient otrzymuje odpowiedź. Odpowiedź zawiera PID procesu partnera.

W tym momencie klient powinien wypisać jednowierszowy komunikat zawierający rozdzielone pojedynczą spacją

- wartość *k*
- wartość *n*
- własny PID
- PID partnera

i zatrzymać się na *s* sekund.

Na końcu klient wysyła na serwer informację o zwolnieniu zasobów, wypisuje komunikat zawierający własny PID i napis KONIEC i kończy się.

Komunikacja

Komunikacja między klientami a serwerem odbywać się może wyłącznie przy pomocy kolejek komunikatów IPC Systemu V. W rozwiązaniu **należy użyć stałej liczby kolejek**.

Dokładny format komunikatów do ustalenia przez studenta.

Zalecam zdefiniowanie kluczy kolejek w pliku nagłówkowym włączanym do programów serwer i klient.

Pozostałe informacje

Rozwiązania (tylko Makefile, pliki źródłowe i opcjonalny plik z opisem) należy nadsyłać za pomocą skryptu submit na adres: solab@mimuw.edu.pl

Ostateczny, nieprzesuwalny termin nadsyłania rozwiązań to: **23 grudnia, 23:59**

W przypadku wątpliwości można zadać pytanie autorowi zadania P. Czarnikowi. Przed zadaniem pytania warto sprawdzić, czy odpowiedź na nie nie pojawiła się na liście często zadawanych pytań.