Projekt 2
Serwer mutexów i condition variables.

Należy rozbudować system Minix o serwer implementujący funkcjonalności 'mutexów' oraz 'condition variables'.

Zarówno mutexy jak i condition variables będą identyfikowane w systemie przez liczby typu `int`. Implementacje opisanych poniżej funkcji powinny zostać dodane do biblioteki systemowej (np. w `/lib/other`).

Mutexy.
Funkcje:

- `mcs_lock(int mutex_id)` - próbuje zarezerwować mutex o numerze przekazanym w argumencie. Jeśli mutex nie jest w posiadaniu żadnego procesu powinien być przydzielony procesowi który wywołał funkcję. W takim przypadku funkcja zwraca 0 (sukces). Jeśli inny proces jest w posiadaniu mutexu bieżący proces powinien być zawieszony aż do momentu kiedy mutex będzie mógł być mu przydzielony lub oczekiwanie na mutex zostanie przerwane sygnałem. W przypadku kiedy proces otrzymuje mutex funkcja zwraca 0 (sukces), w przypadku przerwania sygnałem funkcja zwraca -1 i ustawia `errno` na `EINTR`. Żaden proces nie powinien żądać mutexu który już jest w jego posiadaniu. Zachowanie w takim przypadku jest niezdefiniowane z tym że niedopuszczalna jest sytuacja kiedy wskutek takiego działania przestaje działać system lub serwer mutexów.
- `mcs_unlock(int mutex_id)` - zwalnia mutex o numerze przekazanym w argumencie. Jeśli wołający proces jest w posiadaniu mutexu, funkcja zwraca 0 (sukces) a serwer mutexów przydziela mutex następnemu procesowi z kolejki procesów oczkujących na ten mutex (jeśli kolejka nie jest pusta). Jeśli proces wołający nie jest w posiadaniu tego mutexu funkcja zwróci -1 i ustawi errno na `EPERM`.

Procesy oczekujące na jeden mutex powinny być ustawiane w kolejkę (FIFO).

Condition variables.
Funkcje:

- `mcs_wait(int cond_var_id, int mutex_id)` - zawiesza bieżący proces w oczekiwaniu na zdarzenie identyfikowane przez `cond_var_id`. Proces wywołujący tę funkcję powinien być w posiadaniu mutexu identyfikowanego przez `mutex_id`. Jeśli proces wołający nie posiada odpowiedniego mutexu funkcja powinna zwrócić -1 i ustawić `errno` na `EINVAL`. Jeśli proces wołający jest w posiadaniu mutexu serwer powinien zwolnić mutex i zawiesić wołający proces aż do czasu gdy jakiś inny proces nie ogłosi zdarzenia `cond_var_id` za pomocą funkcji `mcs_broadcast`. W takim przypadku serwer powinien ustawić proces w kolejce procesów oczekujących na mutex `mutex_id` i po otrzymaniu mutexu zwrócić 0 (sukces). Jeśli

czekanie na zdarzenie lub na mutex zostało przerwane sygnałem funkcja powinna zwrócić -1 i ustawić `errno` na `EINTR`. W obu tych przypadkach mutex powinien zostać zwolniony.

- `mcs_broadcast(int cond_var_id)` - ogłasza zdarzenie identyfikowane przez `cond_var_id`. Wszystkie procesy które zawiesiły się w oczekiwaniu na to zdarzenie powinny zostać odblokowane. Każdy z nich po odzyskaniu swojego mutexu powinien zostać wznowiony.

Można przyjąć że w każdym momencie działania serwera co najwyżej 1024 mutex'y są zarezerwowane.

Instrukcja submitowania:

Rozwiązanie będzie testowane w systemie MINIX 3.1.0 (book version). Należy wysłać jedno archiwum zawierające wszystkie pliki źródłowe, które były zmieniane lub dodane. Archiwum będzie rozpakowane w katalogu `/usr/src` instrukcją:

```
gunzip cserv.tar.gz
tar -xf cvserv.tar
```

Po zaktualizowaniu include'ów, rekompilacji bibliotek, serwerów, driverów i obrazu systemu oraz restarcie systemu serwer powinien działać.