Kraków, 29 I 2020

Karmnik i Ring4U
czyli „Symulacja ustawek”
czyli „Grupy zbrojne w sygnały”

(zadanie zaliczeniowe, #2)

1. Wstęp

Projekt składa się z układu dowolnej liczby procesów, wykonujących pięć różnych programów (ewentualnie cztery).
Podstawowy program, `Serwer`, odpowiada za koordynację działań objętych scenariuszem *„Karmnik”*. Współdziałają z nim procesy realizujące `Klientów`. Może być ich wiele i mogą być dodawane w dowolnym momencie. Nie muszą być one spokrewnione z procesem serwera, a komunikują się z nim za pomocą potoków `FIFO` i sygnałów. Klienci tworzą potomków w tempie kontrolowanym przez serwer. Utworzone potomki początkowo trwają w letargu, aż do momentu, gdy `Serwer` nie zmieni trybu działania `Klienta`, który jest ich rodzicem. Od tego momentu `Klient` i jego potomki realizują scenariusz *„Ring4U”*.

Do wykonania scenariusza *„Ring4U”* zawsze przekazywane są pary `Klientów` i oba procesy dostają niezbędne informacje o drugiej stronie. Zmiana scenariusza powoduje, że łagodne potomki przemieniają się w bandy bezwzględnych huliganów, które toczą zażarty bój na śmierć i życie. Ich rodzice stają się przywódcami band i prowodyrami rozruchów. `Klient`, który utraci wszystkie swoje potomki powraca do scenariusza *„Karmnik”*. Drugi z klientów, gdy stwierdzi brak pierwszego, również powraca do *„Karmnik”*, ale dopiero po wykonaniu porządków.

Walki gangów są toczone według reguł, a działania wszystkich huliganów są synchronizowane przez proces `Arbitra` za pomocą `Totemu`.
`Arbiter` jest potomkiem procesu `Serwer`.

2. Karmnik

Na etapie *„Karmnika”*, procesy `Klientów` generują potomków, oraz starają się o skierowanie do drugiego scenariusza. Tempem tworzenia potomków i wyborem procesów do wykonywania *„Ring4U”* steruje `Serwer`.
Podstawowymi kanałami komunikacyjnymi między `Serwerem` i `Klientami` są dwa pliki `FIFO`, o nazwach `FI` oraz `FO`. Za ich istnienie odpowiada `Serwer`, natomiast `Klient`, gdy zajdzie taka konieczność, czeka na ich utworzenie.

Za pomocą pliku `FO`, `Serwer` dostarcza zezwolenia na utworzenie potomka. Natomiast w pliku `FI` uprawnieni klienci wpisują swoje zgłoszenia do uczestniczenia w drugim scenariuszu. Na podstawie tych zgłoszeń `Serwer` wybiera pary procesów, które wraz z potomkami, zmienią scenariusz na *„Ring4U”*.

Zezwolenie jest w postaci jednego bajta, którego `Klienci` starają się odczytać, konkurując przy tym między sobą o dostęp do pliku `FO`. Za każdym razem klientowi wolno pobrać tylko jeden bajt, nawet jeżeli w pliku jest więcej zezwoleń. Po udanym odczycie

z FO, Klient tworzy nowego potomka. Wszystkie potomki są umieszczane w osobnej grupie procesów, różnej od grupy ich rodzica. Potomki nie są agresywne i trwają w letargu, oczekując na zmianę scenariusza. Serwer w równomiernych odstępach czasu przystępuje do wydawania zezwoleń. Ich ilość jest za każdym razem losowana z ustalonego przedziału. Na dodatek, ich wysłanie może zostać zablokowane, jeżeli w danym momencie plik FO nie będzie pusty.

Gdy Klient ma już co najmniej trzech (żywych) potomków, to rozpoczyna starania o zmianę scenariusza. Polega to na wpisywaniu zgłoszenia do pliku FI. Zgłoszenia mają stały rozmiar i format, i zawierają PID procesu oraz identyfikator grupy procesów, w której znajdują się potomki.

Plik FI jest przez większość czasu zamknięty. Serwer otwiera go w nieregularnych odstępach czasu, i to pod warunkiem, że nie ma aktualnie procesów skierowanych do *„Ring4U"*. Plik jest następnie utrzymywany w stanie otwartym przez pewien, stosunkowo niedługi okres. Następnie, przed jego zamknięciem, Serwer pobiera z FI tylko tyle informacji, aby móc wytypować dwóch różnych Klientów. Jeżeli się to powiedzie, procesy te są informowane o zmianie scenariusza.

Ponieważ dostęp do FI i FO jest utrudniony na różne sposoby, Klienty muszą w sposób cykliczny podejmować próby: raz odczytu, a drugi otwarcia i zapisu. Przy czym próby te nie mogą procesu zablokować (przynajmniej na dłuższy czas), ani nie mogą obciążać procesora. Dodatkowo, jeżeli w danym cyklu udał się zapis, to przy najbliżej próbie odczytania FO, Klient rezygnuje z pobrania zezwolenia.

Poinformowanie Klienta, że został wytypowany do drugiego scenariusza polega na wysłaniu do niego sygnału SIGRTMIN+13, wraz z dołączonym identyfikatorem grupy potomków drugiego z klientów. Dodatkowo, z niewielkim opóźnieniem, Serwer informuje Arbitra o uruchomieniu scenariusza. Robi to także za pomocą sygnału SIGRTMIN+13. Uwaga, w czasie, gdy wybrane Klienty i ich potomkowie realizują swoje zadania w ramach *„Ring4U"*, pozostałe procesy (oprócz Arbitra) nadal działają w ramach *„Karmnika"*.

3. Ring4U

Klient, po otrzymaniu sygnału SIGRTMIN+13, zapamiętuje jego nadawcę (PID serwera), a następnie ten sam sygnał, z tymi samymi informacjami kieruje do potomków. W ten sposób przekazuje im identyfikator antagonistycznej grupy procesów.

Potomek, po otrzymaniu SIGRTMIN+13, wybudza się z letargu i podmienia swój kod na program Hooligan. Celem nowego programu jest zwalczenie procesów należących do wskazanej grupy. W swych działaniach kieruje się zasadami ujętymi w *Kodeksie Honorowym Klanu Hooliganów*. Są w nim tylko dwa punkty (więcej huligan nie zapamięta):

a) mordy masowe są zabronione,
b) rytm jest wybijany przez Totem.

Pierwszy z nich oznacza, że atakować można wyłącznie pojedyncze procesy. Ponieważ huligan ma dostarczony identyfikator grupy, musi przed atakiem zlokalizować w niej jakiś jeden proces. Opis tej procedury znajduje się w sekcji "Wybór celu".
Drugi punkt mówi, że operacje bojowe mają być zsynchronizowane z sygnalizacją Totemu. Za istnienie Totemu oraz sterowanie jego sygnalizacją odpowiada proces Arbitra (więcej informacji w sekcji „Arbiter"). Przewidywane są dwa stany Totemu:

1° otwarcie, w czasie którego huligany mają czas na wykonanie dwóch operacji

- zlokalizowanie celu ataku,
- przygotowanie się do obrony,

2° zamknięcie, w którym wykonywany jest jedynie atak.

Ataki polegają na wysyłaniu jednego spośród sygnałów `SIGUSR1` i `SIGUSR2`, za każdym razem losowo wybieranego. `Hooligany` mogą także się bronić poprzez zadeklarowanie własnej obsługi jednego z tych sygnałów (dla drugiego musi być ustanowiona standardowa procedura). Podobnie jak przy ataku, w każdym cyklu sygnał, który ma być broniony, jest losowany.

Walki trwają do momentu, aż po jednej lub obu stronach braknie huliganów. Opis rozpoznania tego stanu oraz dalszych kroków znajduje się w sekcji „Koniec walk".

3.1. Arbiter

`Arbiter` ma za zadanie synchronizować działania walczących `Hooliganów`. Wykorzystuje do tego plik `FIFO` o nazwie `Totem`. Mechanizmem synchronizującym są operacje otwarcia i zamknięcia pliku, którym odpowiadają stany totemu: „otwarcie" i „zamknięcie".

`Arbiter` ma dwa tryby: aktywny i pasywny, przełączane przez Serwer za pomocą sygnału `SIGRTMIN+13`. Po uruchomieniu i wstępnych przygotowaniach, w ramach których tworzony jest plik totemu, `Arbiter` zatrzymuje się w trybie pasywnym. W tym czasie plik totemu ma być w stanie „zamknięty". Po pojawieniu się sygnału, `Arbiter` przechodzi w tryb aktywny, w którym cyklicznie zmienia stan totemu. Natomiast po kolejnym sygnale wraca do trybu pasywnego. Przy czym przejście następuje w najbliższym momencie, w którym `Totem` jest w stanie zamkniętym.

3.2. Wybór celu

Każdy `Hooligan` stara się określić PID jakiegoś procesu, występującego w przeciwnej grupie. W tym celu kieruje pod jej adresem *obelgę*. Członkowie drugiej grupy odpowiadają mu *inwektywami*. `Hooligan` wybiera jedną z inwektyw i zapamiętuje, kto był jej nadawcą.

Realizacja powyższego algorytmu opiera się na wysyłaniu sygnałów. Jako zaczepkę proces wysyła do grupy antagonistycznych procesów sygnał `SIGALRM`. Program `Hooligan` reaguje na taki sygnał pobierając z niego PID nadawcy i odsyłając do niego sygnał `SIGBUS`. Natomiast obsługa `SIGBUS` polega wyłącznie na zapamiętaniu nadawcy (jako celu najbliższego ataku). Dopuszczalne jest, by wielu huliganów wybrało sobie tego samego przeciwnika za cel (*Kodeks Honorowy Klanu Hooliganów* tego nie zabrania).

Uwaga. Ponieważ sygnały nie są kolejkowane, więc mogą być gubione. W konsekwencji może się zmniejszyć pula potencjalnych celów. Dlatego członkowie jednej grupy powinni starać się, by nie wysyłać swoich `SIGALRM` w zbyt krótkim przedziale czasu. (Proste rozwiązanie: wprowadzić niewielkie losowe opóźnienia.)

3.3. Koniec walk

Klienci kontrolują liczbę działających potomków poprzez „zliczanie zębów" (tj. *zombie*). Ten, który stwierdzi, że jego oddział został całkiem wybity, zgłasza się do `Serwera` jako przegrany, wysyłając sygnał `SIGRTMIN+13` z wartością `0`. Następnie samodzielnie powraca do *„Karmnika"*.

Klient, którego przeciwnik przegrał, dowiaduje się o tym fakcie od swoich potomków. Gdy Hooligan stwierdzi, że nie może określić celu następnego ataku, wysyła do herszta bandy (Klienta) sygnał SIGRTMIN+13. Herszt rozpoznaje znaczenie tego sygnału na podstawie jego nadawcy. Gdy Klient już wie, że wygrał, zgłasza ten fakt Serwerowi, wysyłając sygnał SIGRTMIN+13 z wartością 1. Następnie wymienia swoich Hooliganów na spokojniejsze procesy, po czym wraca do scenariusza *„Karmnika"*. Wymiana potomków polega na zabiciu starych i utworzeniu w ich miejsce nowych. Uwaga, nowe potomki muszą należeć do tej samej grupy co stare. (Do wynajdowania żywych Hooliganów można wykorzystać już istniejące procedury obsługi sygnału.)

Serwer, gdy już otrzyma od obu Klientów informację o zakończeniu rozruchów, wprowadza Arbitra w tryb pasywny, wysyłając do niego SIGRTMIN+13.

4. Dalsze szczegóły

4.1. Parametry

Działanie większości programów zależy od wartości parametrów nimi sterujących. Prawie wszystkie parametry będą przekazywane za pomocą zmiennych środowiskowych, dzięki czemu ułatwione będzie zachowanie spójności całego systemu. Wykorzystywane będą następujące zmienne:

CAMP
: ścieżka do katalogu, w którym są umieszczane pliki do komunikacji (FI, FO, Totem) ; używana przez (prawie) wszystkie programy,

RECRUIT_DLY
: czas pomiędzy generowaniem kolejnych zezwoleń (rekrutacje do gangów), używana przez program Serwer,

RECRUIT_MAX
: maksymalna ilość generowanych na raz zezwoleń, Serwer losuje liczbę zezwoleń z przedziału [0,RECRUIT_MAX],

RING_REG
: czas, przez jaki są akceptowane zgłoszenia do uczestnictwa w drugim scenariuszu; technicznie: czas otwarcia pliku FI, używane przez program Serwer,

TOTEM_OP, TOTEM_CL
: czas, przez jaki plik Totem jest wstanie „otwarty" lub „zamknięty" (odpowiednio), używane przez Arbitra.

Jedynym przypadkiem przekazywania parametru jako argumentu uruchomienia, jest program Hooligan, który dostaje identyfikator grupy wrogich procesów.

4.2. Sygnały - podsumowanie

Każdy z programów musi obsługiwać jeden lub kilka sygnałów. Ich działanie i okoliczności wysyłania zostały opisane we wcześniejszych sekcjach. Poniżej znajduje się systematyczne podsumowanie tych informacji.

Serwer

— SIGRTMIN+13 z parametrem liczbowym
: sygnał wysyłany przez Klienta i oznacza jego powrót do *„Karmnika"*, wartość parametru oznacza zwycięstwo (1) lub przegraną (0),

`Klient`
— `SIGRTMIN+13` z parametrem liczbowym
  sygnał wysyłany przez `Serwer`,
  parametr zawiera identyfikator grupy procesów,
— `SIGRTMIN+13`
  sygnał wysyłany przez `Hooligana`, oznaczający koniec walki.

`potomek Klienta`
(może to być osobny program, albo kontynuacja `Klient`)
— `SIGRTMIN+13` z parametrem całkowitym
  wysyłany przez rodzica (`Klient`),
  parametr zawiera identyfikator grupy procesów,

`Hooligan`
(wszystkie sygnały są wysyłane przez inne `Hooligany`)
— `SIGALRM`
  — prowokuje od podpowiedzi: do nadawcy jest odsyłany sygnał `SIGBUS`,
— `SIGBUS`
  — odnotowanie nadawcy sygnału (jako potencjalnego celu przyszłego ataku),
— `SIGUSR1` i `SIGUSR2`
  — sygnały będące wynikiem ataku przez innego `Hooligana`,
  — jeden a nich jest zabójczy, a drugi obsługiwany (wybór losowy).

`Arbiter`
— `SIGRTMIN+13`
  — wysyłany przez `Serwer`,
  — powoduje zmianę tryby działania (aktywny–pasywny).