

Opis projektu 2:

Symulacja lotniska przy użyciu mechanizmów systemowych

1. Wstęp

Celem projektu jest zasymulowanie sieci lotnisk, ze szczególnym naciskiem położonym na użycie mechanizmów systemowych. Dla większej czytelności użyto technik obiektowych języka C++. Dzięki takiemu rozwiązaniu całość cechuje się dużą skalowalnością. Wykorzystane techniki powodują praktycznie bezproblemowe przenoszenie pomiędzy systemami zgodnymi z normą POSIX, które zawierają bibliotekę do obsługi wątków *pthread*.

2. Opis działania

Aby zaprezentować użycie mechanizmów systemowych utworzone zostały 3 w dużej mierze niezależne programy:

- ❖ Creator
- ❖ Lotnisko
- ❖ Samolot

Zadaniem **kreatora** jest tworzenie nowych procesów samolotów. Może on generować je w sposób losowy z zadanymi parametrami dotyczącymi odstępu między kolejnymi generowaniami, wielkością samolotu (czas lądowania) oraz czas oczekiwania na lotnisku. Drugim sposobem jest generowanie według wcześniej ustalonego planu przylotów i odlotów. Możliwe jest wtedy ustalenie czasu startu/lądowania i wielkości samolotu priorytetu i lotnisk, na których ma lądować dany samolot.

Zakończenie pracy kreatora następuje po odebraniu sygnału QUIT. Jeśli sygnał jest odbierany po raz pierwszy to następuje przerwanie generowania samolotów i oczekiwanie na zakończenie lotu przez wszystkie wygenerowane (wyświetlany jest postęp operacji). Po odebraniu drugiego sygnału QUIT bezwarunkowo jest kończony proces kreatora, jednak samoloty dalej lądują i startują.

Lotnisko jest programem-serwerem – tworzy kolejkę komunikatów, przez którą odbiera zgłoszenia (żądania lądowania i startu). Klucz kolejki jest generowany na podstawie identyfikatora podanego jako argument z linii komend (jest to wymagany argument). Żądania są odbierane według priorytetów, mimo że do dłuższego oczekiwania w kolejce może dojść tylko w sytuacji ekstremalnego przeciążenia lotniska. Wszystkie żądania są cały czas odbierane z kolejki komunikatów i przesyłane do wewnętrznej synchronizowanej kolejki priorytetowej. Bezpośrednio po odebraniu następuje wysłanie odpowiedzi przez prywatną kolejkę fifo, której identyfikator jest przesyłany razem z żądaniem lądowania/startu z informacją dotyczącą przewidywanego czasu oczekiwania.

Z powyższej kolejki czytają wątki – zarządcy kolejnych pasów startowych. Odczyt następuje, gdy dany pas jest wolny i może obsłużyć lądujący lub startujący samolot.

Dostęp do kolejki jest synchronizowany za pomocą muteksów i semaforów, a o kolejności odczytu decyduje priorytet samolotu. Po odczytaniu z kolejki wysyłana jest zgoda poprzez prywatną kolejkę fifo na lądowanie lub start. Po zakończeniu lądowania/startu jest wysyłane również potwierdzenie zakończenia operacji.

Przebieg pracy jest na bieżąco wyświetlany przy użyciu biblioteki *ncurses* – monitorowany jest stan pasa oraz wyświetlana jest historia żądań wysyłanych do lotniska oraz przebieg obsługi samolotów na danych pasach.

W wyniku pracy lotniska generowane są statystyki dotyczące lotniska (średni czas oczekiwania, wariancja średniego czasu oczekiwania, ilość żądań lądowań/ startów...) oraz statystyki dotyczące czasu pracy poszczególnych pasów (ilość obsłużonych lądowań/startów, średni czas lądowania, wariancja średniego czasu lądowania...). Wszystkie statystyki są zapisywane w pliku *lotnisko-stats*.

Podczas pracy serwera jest tworzony – dopisywany - plik logu (z aktualnym czasem): *lotnisko.log*.

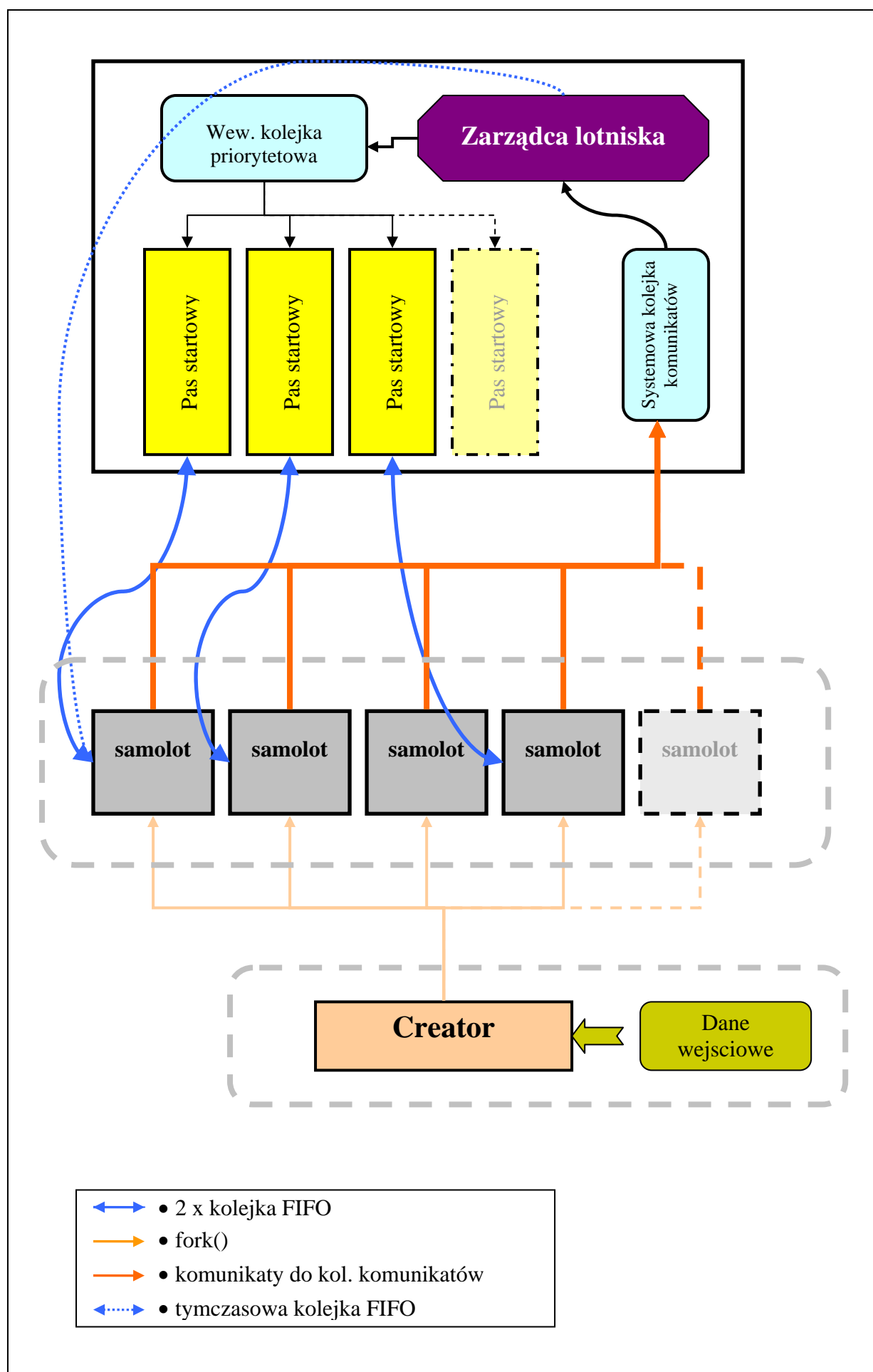
Zakończenie pracy lotniska następuje po wysłaniu sygnału: TERM, QUIT. Po odebraniu ww. sygnałów proces lotniska kończy prace usuwając kolejkę komunikatów.

Proces **samolotu** może być uruchamiany bezpośrednio z zadanymi parametrami priorytetu, nazwy samolotu, czasu oczekiwania i lądowania, jednak najczęściej jest on wywoływany przez proces kreatora. Po uruchomieniu następuje wysłanie utworzenie kolejki fifo oraz wysłanie jej identyfikatora wraz z żądaniem lądowania. Po odebraniu potwierdzenia otrzymania żądania wraz z przewidywanym czasem oczekiwania następuje oczekiwanie na zezwolenie lądowania (wysyłane przez zarządcę konkretnego pasa startowego). Wówczas pas jest zajmowany na zadana liczbę sekund (w zależności od wielkości samolotu). Po zakończeniu lądowania samolot przechodzi w stan oczekiwania (na lotnisku), po czym wysyła żądanie startu i procedura rozpoczyna się od początku. Następnie, jeśli z linii komend podano lotnisko jako argumentu to procedura startu/lądowania jest powtarzana dla kolejnych lotnisk.

3. Mechanizmy systemowe

Użyte w projekcie mechanizmy systemowe to:

- Kolejka fifo – prywatna komunikacja z samolotami
- Kolejki komunikatów – ogólnie dostępna kolejka do przesyłania wiadomości do serwera.
- Semaforey
- Wątki – zarządcy pasów, synchronizacja między wątkami za pomocą muteksów.
- Muteksy
- Synchronizowana wewnętrzna kolejka priorytetowa
- Użycie biblioteki *ncurses*



4. Możliwości rozszerzenia i ograniczenia

- Nieograniczona możliwość rozbudowy prowadzonych statystyk – istnieje specjalna klasa *Status*, która „przyjmuje” dane statystyczne i pod koniec pracy lotniska generuje plik ze statystykami.
- Liczba pasów przekazywana przez parametr.
- Możliwość zachodzenia zjawisk losowych – mgła itd.
- Bogatsze środowisko do obserwacji symulacji – stosunkowo łatwe rozszerzenie dzięki stworzeniu klas *Display* odpowiedzialnej za wyświetlanie oraz dzięki użyciu biblioteki *ncurses*.

