

Systemy operacyjne 2

Problem ucztujących filozofów

Piotr Józefek 272311

1. Wersja z zakleszczeniem

Ta strategia implementuje rozwiązanie problemu uczujących filozofów, w którym każdy filozof próbuje najpierw podnieść lewą, a następnie prawą pałeczkę. Takie podejście prowadzi do potencjalnego zakleszczenia, gdy wszyscy filozofowie podniosą swoją lewą pałeczkę i czekają na prawą która jest już trzymana przez sąsiada.

```
Symulacja Problemu Pieciu Filozofow wersja 1
Filozof 1: Głodny (Zjadl: 0)
Filozof 2: Głodny (Zjadl: 0)
Filozof 3: Głodny (Zjadl: 0)
Filozof 4: Głodny (Zjadl: 0)
Filozof 5: Głodny (Zjadl: 0)

Stan pałeczek:
Pałeczka 1: Używana przez Filozofa 1
Pałeczka 2: Używana przez Filozofa 2
Pałeczka 3: Używana przez Filozofa 3
Pałeczka 4: Używana przez Filozofa 4
Pałeczka 5: Używana przez Filozofa 5
```

2. Wersja z zagłodzeniem

Ta strategia implementuje podejście mające na celu zademonstrowanie zjawiska zagłodzenia wątku. Dla ostatniego filozofa występuje odwrotna kolejność wyboru pałeczek, najpierw blokując prawa, a następnie, z opóźnieniem, próba wzięcia lewej pałeczki. W przypadku niepowodzenia zajęcia lewej pałeczki filozof zwalnia posiadana pałeczkę i przechodzi w stan myślenia.

```
Symulacja Problemu Pieciu Filozofow wersja 2
Filozof 1: Je (Zjadl: 112)
Filozof 2: Mysli (Zjadl: 128)
Filozof 3: Mysli (Zjadl: 132)
Filozof 4: Je (Zjadl: 136)
Filozof 5: Głodny (Zjadl: 64)

Stan pałeczek:
Pałeczka 1: Używana przez Filozofa 1
Pałeczka 2: Używana przez Filozofa 1
Pałeczka 3: Wolna
Pałeczka 4: Używana przez Filozofa 4
Pałeczka 5: Używana przez Filozofa 4
```

3. Wersja bez zakleszczenia i zagłodzenia

Ta strategia zapobiega zakleszczeniu poprzez ograniczenie liczby filozofów, którzy mogą jednocześnie próbować jeść do liczby pałeczek - 1. Dodatkowo używa zmiennej warunkowej, aby zatrzymać filozofów, gdy zostanie osiągnięte maksimum jedzących filozofów. Problem zakleszczeń i cyklicznych zależności został wyeliminowany poprzez podnoszenie pałeczek w ustalonej kolejności zaczynając od najniższego indeksu.

```
Symulacja Problemu Pieciu Filozofow wersja 3
Filozof 1: Mysli (Zjadl: 131)
Filozof 2: Mysli (Zjadl: 133)
Filozof 3: Glodny (Zjadl: 134)
Filozof 4: Je (Zjadl: 140)
Filozof 5: Mysli (Zjadl: 127)

Stan paleczek:
Paleczka 1: Wolna
Paleczka 2: Wolna
Paleczka 3: Uzywana przez Filozofa 3
Paleczka 4: Uzywana przez Filozofa 4
Paleczka 5: Uzywana przez Filozofa 4
```

4. Kod

```
void filozof(int id, int strategia, std::vector<std::string>& stan_filozofow)
{
    while (true)
    {
        int lewa_paleczka = id;
        int prawa_paleczka = (id + 1) % NUM_PALECZEK;

        stan_filozofow[id] = "Głodny";
        wyswietl_stan(stan_filozofow, strategia);

        if (strategia == 1) // Wersja z zakleszczeniem
        {
            paleczki[lewa_paleczka].lock();
            {
                std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
                stan_paleczek[lewa_paleczka] = id + 1;
            }
            wyswietl_stan(stan_filozofow, strategia);
            usleep(300 * 1000); // 300 milisekund

            paleczki[prawa_paleczka].lock();
            {
                std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
                stan_paleczek[prawa_paleczka] = id + 1;
            }

            stan_filozofow[id] = "Je";
            {
                std::lock_guard<std::mutex> lock(mutex_liczba_posilkow);
                liczba_posilkow[id]++;
            }
            wyswietl_stan(stan_filozofow, strategia);
            usleep(losuj_czas(czas_jedzenia) * 1000);

            {
                std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
                stan_paleczek[lewa_paleczka] = 0;
                stan_paleczek[prawa_paleczka] = 0;
            }
            paleczki[lewa_paleczka].unlock();
            paleczki[prawa_paleczka].unlock();
        }
    }
}
```

```

else if (strategia == 2) // Wersja z zaglodzeniem
{
    if (id == NUM_FILOZOFOW - 1)
    {
        paleczki[prawa_paleczka].lock();
        {
            std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
            stan_paleczek[prawa_paleczka] = id + 1;
        }
        usleep(50 * 100 * 1000);

        if (!paleczki[lewa_paleczka].try_lock())
        {
            paleczki[prawa_paleczka].unlock();
            usleep(losuj_czas(czas_myslenia) * 1000);
            continue;
        }
        {
            std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
            stan_paleczek[lewa_paleczka] = id + 1;
        }
    }
    else
    {
        paleczki[lewa_paleczka].lock();
        {
            std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
            stan_paleczek[lewa_paleczka] = id + 1;
        }
        usleep(50 * 1000);
        paleczki[prawa_paleczka].lock();
        {
            std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
            stan_paleczek[prawa_paleczka] = id + 1;
        }
    }

    stan_filozofow[id] = "Je";
    {
        std::lock_guard<std::mutex> lock(mutex_liczba_posilkow);
        liczba_posilkow[id]++;
    }
    wyswietl_stan(stan_filozofow, strategia);
    usleep(losuj_czas(czas_jedzenia) * 1000);

    {
        std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
        stan_paleczek[lewa_paleczka] = 0;
        stan_paleczek[prawa_paleczka] = 0;
    }
    paleczki[lewa_paleczka].unlock();
    paleczki[prawa_paleczka].unlock();
}

```

```

else if (strategia == 3) // Wersja z zabezpieczeniami
{
    std::unique_lock<std::mutex> lock_liczba(mutex_liczba_jedzacych);
    cv_liczba_jedzacych.wait(lock_liczba, [&] { return liczba_jedzacych < NUM_FILOZOFOW - 1; });
    liczba_jedzacych++;
    lock_liczba.unlock();

    int pierwsza_paleczka = std::min(lewa_paleczka, prawa_paleczka);
    int druga_paleczka = std::max(lewa_paleczka, prawa_paleczka);

    paleczki[pierwsza_paleczka].lock();
    {
        std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
        stan_paleczek[pierwsza_paleczka] = id + 1;
    }
    paleczki[druga_paleczka].lock();
    {
        std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
        stan_paleczek[druga_paleczka] = id + 1;
    }

    stan_filozofow[id] = "Je";
    {
        std::lock_guard<std::mutex> lock(mutex_liczba_posilkow);
        liczba_posilkow[id]++;
    }
    wyswietl_stan(stan_filozofow, strategia);
    usleep(losuj_czas(czas_jedzenia) * 1000);

    {
        std::lock_guard<std::mutex> lock_stan(mutex_stan_paleczek);
        stan_paleczek[pierwsza_paleczka] = 0;
        stan_paleczek[druga_paleczka] = 0;
    }
    paleczki[pierwsza_paleczka].unlock();
    paleczki[druga_paleczka].unlock();

    std::lock_guard<std::mutex> lock_liczba_zwolnienie(mutex_liczba_jedzacych);
    liczba_jedzacych--;
    cv_liczba_jedzacych.notify_one();
}

stan_filozofow[id] = "Mysli";
wyswietl_stan(stan_filozofow, strategia);
usleep(losuj_czas(czas_myslenia) * 1000);
}
}

```

5. Całość kodu

https://github.com/Piotrwroc/SO2_filozofowie