PPaDS-Zadanie 4 Jakub Trstenský 92378

1. Urobte analýzu, o aké typy synchronizačných úloh sa v tejto úlohe jedná

Jedná sa o synchronizačný typ vzájomného vylúčenia kategórii. Senzory tvoria prvú kategóriu, operátori tvoria druhú skupinu. Obidve kategórie môžu naraz pristupovať k údajom a to kvôli tomu, že operátori iba čítajú dáta zapísané senzormi, nijak ich nemodifikujú. Môžu teda sprístupňovať údaje zo senzorov naraz. Zároveň aj senzory môžu naraz zapisovať, pretože každý zapisuje do svojho vlastného priestoru, takže sa neblokujú. Taktiež je použitá priorita kategórie procesov, aby sme uprednostnili zápis senzorov, aby mali operátori stále aktuálne dáta.

2. Presne namapujte vami zvolené synch. úlohy na jednotlivé časti zadania

Vzájomné vylúčenie kategórii – som použil preto, lebo chcel som zamädziť prístup senzorom počas čítania dát monitormi a naopak, keď senzory zapisovali, aby tam nečítali monitory.

Priorita kategórie procesov – použil som na senzory, aby mali prioritu zápisu dát. Pretože monitory môžu stále čítať a chcú aktuálne dáta. Čiže v prípade ak majú senzory aktuálne dáta majú priorutu, počkajú kým dočítajú monitory a ihneď zapíšu aktuálne dáta

Bariéra – Bariéra je použitá na začiatku, aby monitory mohli čítať nejaké dáta. V programe idú prioritne vždy prvé senzory, aby zapísali dáta a následne už môžu čítať monitory zapísané dáta.

3. Pseudokód

```
class Operator:
    def __init__():
        lightswitch_operators = LightSwitch()
        without_operators = Semaphore(1)
        without_sensors = Semaphore(1)
        # inicializovanie semaforu na 3 kvoli poctu senzorom 3
        simple_barrier = SimpleBarrier(3)

    def operator(operator_id):
        while True:
            # bariera, na ktorej cakaju operatori(monitory) pretoze najskor musia cidla zapisat data
            simple_barrier.wait()

            #cakanie monitorov, kym senzory zapisu data
            without_operators.wait()

            # ziskanie pristupu k datam
            num_reading_ops = self.ls_ops.lock(self.without_sensors)

            # monitory uvolni pristup dalsiemu monitoru ku citaniu (turniket)
            without_operators.signal()

            # proces citania monitorov
            print(
                'monit "%03d": pocet_citajucich_monitorov=%03d,'
                'trvanie_citania=%0.3f\n'
                % (operator_id, num_reading_ops, waiting_time_of_operator)
            )
            sleep(40-50 ms)

            # monitory docitali a davaju pristup senzorom opat zapisovat
            self.ls_ops.unlock(self.without_sensors)
```

```
class Sensor:
    def __init__():
        lightswitch_sensors = LightSwitch()
        written_data = Event()
        without_sensors = Semaphore(1)
        without_operators = Semaphore(1)
        simple_barrier = SimpleBarrier(3)

    def sensor_P_T(sensor_id):
        while True:

            #trvanie senzoru kym nabehne
            sleep(randint(50-60 ms)

            # zamadzenie pristupu pre operatorov (monitory) pocas zapisovania
            num_writing_sens = self.ls_sens.lock(self.without_operators)

            # cakanie senzorov kym monitory nedocitaju
            self.without_sensors.wait()

            # uvolnenie vstupu pre dalsi senzor (turniket)
            self.without_sensors.signal()


            # Vykonavanie zapisovania senzorom
            print(
                'cidlo "%03d": pocet_zapisujucich_cidiel=%03d,'
                "trvanie_zapisu=%0.3f\n"
                % (sensor_id, num_writing_sens, waiting_time_of_sensor)
            )
            sleep(10-20 ms)

            # bariera, aby prvy krat zapisali senzory data este predtym ako budu monitory citat
            self.simple_barrier.wait()

            # uvolnenie miestnosti na citanie pre monitory
            self.ls_sens.unlock(self.without_operators)
```

```
def sensor_H(sensor_id):
    while True:

        #trvanie senzoru kym nabehne
        sleep(randint(50-60 ms)

        # zamadzenie pristupu pre operatorov (monitory) pocas zapisovania
        num_writing_sens = self.ls_sens.lock(self.without_operators)

        # cakanie senzorov kym monitory nedocitaju
        self.without_sensors.wait()

        # uvolnenie vstupu pre dalsi senzor (turniket)
        self.without_sensors.signal()

        # Vykonavanie zapisovania senzorom
        print(
            'cidlo "%03d": pocet_zapisujucich_cidiel=%03d,'
            "trvanie_zapisu=%0.3f\n"
            % (sensor_id, num_writing_sens, waiting_time_of_sensor)
        )
        sleep(20-25 ms)

        # bariera, aby prvy krat zapisali senzory data este predtym ako budu monitory citat
        self.simple_barrier.wait()

        # uvolnenie miestnosti na citanie pre monitory
        self.ls_sens.unlock(self.without_operators)
```