# Zistite, o akú kombináciu synchronizačných problémov sa jedná

Ide o úlohu Hodujúcich divochov, ktorá je založená na myšlienke Producentov a konzumentov, rozšírená o sledovanie stavu „skladu“ (koľko porcií sa nachádza v hrnci).

Štandardné riešenie Hodujúcich divochov však treba rozšíriť ešte o bariéru, aby všetci kuchári dokončili svoju časť prípravy jedla, a následne iba jeden z nich celé jedlo pridal do hrnca a signalizoval divochom že môžu jesť.

V mojom riešení počítam s tým, že všetci kuchári spolupracujú na tvorbe jednej dávky M porcií. V prípade že divoch zistí že už nie je porcia, nabije semafor emtyPot hodnotou C, teda dovolí všetkým vláknam kuchárov vykonávať kód. Kód kuchára najprv volá funkciu ktorá simuluje vykonanie svojej časti práce na celkovom jedle. Nasleduje volanie bariéry inicializovanej na hodnotu C – počet kuchárov. Tam následne čakajú na to kým posledný neprejde do bariéry, ten pridá jedlo ako celok ( M porcií), povolí ostatým opustiť bariéru a signalizuje divochom že je hrniec plný.

def init():

    mutex := Mutex()

    servings := 0

    fullPot := Semaphore(0)

    emptyPot := Semaphore(0)

    barrier1 := SimpleBarrier()

    barrier2 := SimpleBarrier()

cooks\_barrier := CooksBarrier(C)

    for savage\_id in [0, 1, 2, ..., N-1]:

        create\_and\_run\_thread(savage, savage\_id)

for cook\_id in [0, 1, 2, ..., C-1]:

        create\_and\_run\_thread(cook, cook\_id)

def getServingFromPot(savage\_id):

    print("divoch %2d: beriem si porciu", savage\_id)

    servings := servings - 1

def savage(savage\_id):

    while True:

        barrier1.wait("divoch %2d: prisiel som na veceru, uz nas je %2d",

                      savage\_id,

                      print\_each\_thread = True)

        barrier2.wait("divoch %2d: uz sme vsetci, zaciname vecerat",

                      savage\_id,

                      print\_last\_thread = True)

        mutex.lock()

        print("divoch %2d: pocet zostavajucich porcii v hrnci je %2d" % (savage\_id, servings))

        if servings == 0:

            print("divoch %2d: budim kuchara" % savage\_id)

            emptyPot.signal(C)

            fullPot.wait()

        getServingFromPot(savage\_id)

        mutex.unlock()

        # konkurentne vykonavany kod

        print("divoch %2d: hodujem" % savage\_id)

def prepare\_part\_of\_meal (id):

    print("kuchar %2d : pripravuje svoju časť", id

def cook():

    while True:

        emptyPot.wait()

        prepare\_part\_of\_meal(cook\_id)

        cooks\_barrier.cooks\_wait(cook\_id, shared)

// trieda CooksBarrier vychádza z klasickej Simple Bariéry, metódu wait upravuje pre potreby // zadania na metódu cooks\_wait

class CooksBarrier():

//standard parameters

def cooks\_wait(id, shared):

mutex.lock()

counter += 1

print(„kuchar %2d dokončil svoju časť“, (id))

if counter == N:

counter := 0

// ak všetci kuchári dokončili svoju časť tak sa jedlo ako celok pridá

shared.servings +=M

print(„kuchar %2d pridal M porcií“, (id))

// pustime všetkých kuchárov von z bariéry

sem.signal(C)

// signalizujeme divochom že majú jedlo hotové

sahred.full\_pot\_signal()

mutex.unlock()

sem.wait()