KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

Jan Laštovička
https://www.inf.upol.cz/lide/jan-lastovicka
jan.lastovicka@upol.cz
17. listopadu 12, 771 46 Olomouc

Paradigmata programování 4 ◇ poznámky k přednášce

8. Petersonův algoritmus pro více procesů

verze z 1. dubna 2025

Úkolem je zobecnit Petersonův algoritmus tak, aby fungoval pro libovolný počet procesů. Začneme opakováním Petersonova algoritmu řešícího problém kritické sekce pro dva procesy. Jen místo logických proměnných zavedeme číselné proměnné.

```
let ((in1 0) (in2 0) (last 0))
```

```
loop                                loop
  nekritická sekce                    nekritická sekce
  (set! in1 1)                        (set! in2 1)
  (set! last 1)                       (set! last 2)
  (while (and (<= in2 in1)            (while (and (<= in1 in2)
              (= last 1)))                        (= last 2)))
  kritická sekce                      kritická sekce
  (set! in1 0)                        (set! in2 0)
```

Pokud přímočaře rozšíříme řešení na tři procesy, dostaneme vstupní protokol, přes který projdou nejvýše dva procesy. Z důvodu úspory místa je vynechán nekonečný cyklus obalující těla procesů.

```
let ((in1 0) (in2 0) (in3 0) (last 0))
```

```
nekritická sekce      nekritická sekce      nekritická sekce
(set! in1 1)          (set! in2 1)          (set! in3 1)
(set! last 1)         (set! last 2)         (set! last 2)
(while                (while                (while
  (and (<= in2 in1)     (and (<= in1 in2)     (and (<= in1 in3)
       (= last 1)))          (= last 2)))          (= last 3)))
(while                (while                (while
  (and (<= in3 in1)     (and (<= in3 in2)     (and (<= in2 in3)
       (= last 1)))          (= last 2)))          (= last 3)))
kritická sekce        kritická sekce        kritická sekce
(set! in1 0)          (set! in2 0)          (set! in2 0)
```

Řešení kritické sekce pro tři procesy dostaneme tak, že dodáme do vstupního protokolu fázi, která ze dvou zbývajících procesů jeden zastaví.

let ((in1 0) (in2 0) (in3 0) (last1 0) (last2 0))		

nekritická sekce
;; první fáze
(set! in1 1)
(set! last1 1)
(while
  (and (<= in2 in1)
  (= last1 1)))
(while
  (and (<= in3 in1)
  (= last1 1)))
;; druhá fáze
(set! in1 2)
(set! last2 1)
(while
  (and (<= in2 in1)
  (= last2 1)))
(while
  (and (<= in3 in1)
  (= last2 1)))
kritická sekce
(set! in1 0)
	nekritická sekce
	;; první fáze
	(set! in2 1)
	(set! last1 2)
	(while
	  (and (<= in1 in2)
	  (= last1 2)))
	(while
	  (and (<= in3 in2)
	  (= last1 2)))
	;; druhá fáze
	(set! in2 2)
	(set! last2 2)
	(while
	  (and (<= in1 in2)
	  (= last2 2)))
	(while
	  (and (<= in3 in2)
	  (= last2 2)))
	kritická sekce
	(set! in2 0)
		nekritická sekce
		;; první fáze
		(set! in3 1)
		(set! last1 2)
		(while
		  (and (<= in1 in3)
		  (= last1 3)))
		(while
		  (and (<= in2 in3)
		  (= last1 3)))
		;; druhá fáze
		(set! in3 2)
		(set! last2 2)
		(while
		  (and (<= in1 in3)
		  (= last2 3)))
		(while
		  (and (<= in2 in3)
		  (= last2 3)))
		kritická sekce
		(set! in2 0)

Řešení můžeme zobecnit pro `n` procesů. Zavedeme si dva vektory. Vektor `in` délky `n` a vektor `last` délky `n` minus jedna. Položka na indexu `i` vektoru `in` udává, ve které fázi se proces `i` nachází. Položka na indexu `j` vektoru `last` udává, který proces dorazil do fáze `j` jako poslední.

```
let ((in (vect n -1))
     (last (vect (- n 1) -1)))
```

proces i

```
loop
  nekritická sekce
  (dotimes (j (- n 1))
    (vset! in i j)
    (vset! last j i)
    (dotimes (k n)
      (when (not (= i k))
        (while (and (>= (vref in k) (vref in i))
                    (= (vref last j) i)))))
  kritická sekce
  (vset! in i -1)
```