

Systemy operacyjne

## Współbieżność i synchronizacja procesów

Dariusz Wawrzyniak

UCZELNIA  
ONLINE

Systemy operacyjne



### Podstawowe definicje i oznaczenia

- $C$  — zbiór instrukcji (operacji), możliwych do wykonania przez jednostkę przetwarzającą (np. dodawanie, odejmowanie, skok, rozgałęzienie warunkowe)
- $D$  — zbiór danych, przetwarzanych (modyfikowanych lub czytanych) w ramach wykonywania operacji przez jednostkę przetwarzającą
- $e_i(c, O)$  — wykonanie w ramach procesu  $P_i$  instrukcji atomowej  $c \in C$  na operandach ze zbioru  $O \subseteq D$

Współbieżność i synchronizacja procesów (4)

Systemy operacyjne



### Plan wykładu

- Abstrakcja programowania współbieżnego
- Instrukcje atomowe i ich przeplot
- Istota synchronizacji
- Kryteria poprawności programów współbieżnych
- Klasyfikacja mechanizmów synchronizacji
- Wzajemne wykluczania
- Algorytmy wzajemnego wykluczania (alg. Petersona i Lamporta)
- Złożone instrukcje atomowe (**test&set**, **exchange**)

Współbieżność i synchronizacja procesów (2)

Systemy operacyjne



### Stan procesu i zdarzenie

- Stan procesu obejmuje te elementy (wartości zmiennych, rejestrów, stan zasobów), które między innymi determinują następną instrukcję do wykonania.
- Wykonanie instrukcji określane jest jako **zdarzenie** lub **akcja**.
- Zbiór instrukcji do wykonania oraz zależności pomiędzy nimi określone są przez program procesu.
- Zajście zdarzenia zdeterminowane jest zatem przez program oraz bieżący stan procesu.

Współbieżność i synchronizacja procesów (5)

Systemy operacyjne



### Wprowadzenie do abstrakcji przetwarzania współbieżnego

- Realizacja przetwarzania polega na wykonywaniu instrukcji przez jednostkę przetwarzającą (procesor).
- Instrukcje wykonywane są w kontekście jakiegoś procesu.
- Wykonanie instrukcji (akcja) oznacza zajście zdarzenia w procesie, skutkiem czego jest zmiana stanu procesu.
- Zajście zdarzenia jest zdeterminowane poprzedzającym je stanem procesu.
- Na stan procesu wpływają pośrednio akcje innych procesów w systemie, przejawiające się w stanie współdzielonych zasobów.

Współbieżność i synchronizacja procesów (3)

Systemy operacyjne



### Proces sekwencyjny

- Proces (wątek) sekwencyjny jest wykonaniem ciągu instrukcji, opisanych przez program dla tego procesu (procedurę dla wątku), w taki sposób, że następna akcja nie rozpocznie się, zanim nie skończy się poprzednia.
- Zdarzenie (akcja) w procesie  $P_i$  oznacza zmianę stanu tego procesu, co formalnie opisane jest poprzez odwzorowanie (przejście, tranzycję):  
 $L: S_i \times E_i \rightarrow S_i$ , gdzie  
 $S_i$  — zbiór stanów procesu  $P_i$   
 $E_i$  — zbiór zdarzeń w procesie  $P_i$

Współbieżność i synchronizacja procesów (6)

Systemy operacyjne

**Relacja lokalnego porządku**



- Proces  $P_i$  jest ciągiem (skończonym lub nieskończonym) następujących po sobie naprzemiennie stanów i zdarzeń  $s_i^0, e_i^1, s_i^1, e_i^2, s_i^2, \dots$ , gdzie:
  - $s_i^0$  — stan początkowy procesu  $P_i$
  - $\forall_{k \geq 0} s_i^k \in S_i$
  - $\forall_{k > 0} e_i^k \in E_i$
  - $\forall_{k \geq 0} L(s_i^k, e_i^{k+1}) = s_i^{k+1}$
- Relację porządku w procesie  $P_i$  — odzwierciedlającą kolejność stanów i zdarzeń w ciągu — nazywana będzie lokalnym porządkiem i oznaczana symbolem  $\rightarrow_i$

Współbieżność i synchronizacja procesów (7)

Systemy operacyjne

**Niedeterminizm przetwarzania**



- Zdarzenie, które może pojawić się w określonym stanie przetwarzania, określane jest jako zdarzenie dopuszczalne.
- W przetwarzaniu współbieżnym z każdym sekwencyjnym procesem w danym stanie związane jest jedno zdarzenie. W stanie całego przetwarzania jest zatem zbiór zdarzeń dopuszczalnych.
- W zależności od dostępności zasobów (procesora, magistrali systemowej) oraz decyzji planisty, wykonywany będzie jeden z procesów gotowych. Należy więc przyjąć, że zajście jednego ze zdarzeń dopuszczalnych ma charakter losowy.

Współbieżność i synchronizacja procesów (10)

Systemy operacyjne

**Współbieżna realizacja zbioru procesów**



- Zdarzenie w systemie współbieżnym, złożonym z procesów sekwencyjnych  $P_1, P_2, \dots, P_n$ , oznacza zajście zdarzenia w jednym z procesów.
- Zdarzenie, zmieniając stan jednego procesu, zmienia stan całego systemu, co formalnie opisane jest poprzez odwzorowanie (przejście, tranzycję):
  - $G: \Sigma \times \Lambda \rightarrow \Sigma$ , gdzie
  - $\Sigma \subseteq S_1 \times S_2 \times \dots \times S_n$  — zbiór stanów systemu
  - $\Lambda = E_1 \cup E_2 \cup \dots \cup E_n$  — zbiór zdarzeń w systemie

Współbieżność i synchronizacja procesów (8)

Systemy operacyjne

**Przeplot i osiągalność stanu**



- Przeplotem jest zbiór zdarzeń  $\Lambda$ , uporządkowany przez relację globalnego porządku  $\rightarrow$ , spełniającą następujący warunek:
 
$$\forall e, e' \in \Lambda \exists_i e \rightarrow_i e' \Rightarrow e \rightarrow e'$$
- Stan  $\sigma'$  systemu jest osiągalny ze stanu  $\sigma$ , co będzie oznaczone przez  $\sigma \rightsquigarrow \sigma'$ , jeśli zachodzi jeden z warunków:
  - są to te same stany, czyli  $\sigma \equiv \sigma'$
  - $\exists_{e \in \Lambda} G(\sigma, e) = \sigma'$
  - $\exists_{\sigma' \in \Sigma} \sigma \rightsquigarrow \sigma'' \wedge \sigma'' \rightsquigarrow \sigma'$

Współbieżność i synchronizacja procesów (11)

Systemy operacyjne

**Relacja globalnego porządku**



- Przetwarzanie współbieżne zbioru procesów sekwencyjnych  $P_1, P_2, \dots, P_n$ , jest ciągiem (skończonym lub nieskończonym) następujących po sobie naprzemiennie stanów systemu i zdarzeń  $\sigma^0, e^1, \sigma^1, e^2, \sigma^2, \dots$ , gdzie:
  - $\sigma^0$  — stan początkowy  $\langle s_1^0, s_2^0, \dots, s_n^0 \rangle$
  - $\forall_{k \geq 0} \sigma^k \in \Sigma$
  - $\forall_{k > 0} e^k \in \Lambda$
  - $\forall_{k \geq 0} G(\sigma^k, e^{k+1}) = \sigma^{k+1}$
- Relacja porządku w systemie — odzwierciedlająca kolejność stanów i zdarzeń w ciągu — nazywana będzie globalnym porządkiem i oznaczana symbolem  $\rightarrow$

Współbieżność i synchronizacja procesów (9)

Systemy operacyjne

**Procesy niezależne a procesy współpracujące**



- Niech  $D_i \subseteq D$  oznacza zbiór danych przetwarzanych przez proces  $P_i$ , czyli dla ciągu instrukcji procesu  $P_i$ :  $e_i^1(c_i^1, O_i^1), e_i^2(c_i^2, O_i^2), \dots$ 

$$D_i = \bigcup_k O_i^k$$
- Dwa procesy,  $P_i$  i  $P_j$ , są **niezależne**, jeśli
 
$$D_i \cap D_j = \emptyset$$
- Dwa procesy,  $P_i$  i  $P_j$ , **współpracują** (są w interakcji), jeśli
 
$$D_i \cap D_j \neq \emptyset$$

Współbieżność i synchronizacja procesów (12)

Systemy operacyjne

**Dane współdzielone a lokalne**

- Dane lokalne (prywatne) procesu  $P_i$  to takie, które są przetwarzane wyłącznie przez  $P_i$ , formalnie zatem jest to zbiór:
 
$$D_i \setminus \bigcup_{1 \leq j \leq n, j \neq i} D_j$$
- Dane współdzielone przez proces  $P_i$  to takie, które przetwarzane są oprócz  $P_i$  przez co najmniej jeszcze jeden inny proces, formalnie zatem jest to zbiór:
 
$$D_i \cap \bigcup_{1 \leq j \leq n, j \neq i} D_j$$

Współbieżność i synchronizacja procesów (13)

Systemy operacyjne

**Przykład przeplotu instrukcji RISC**

|                     | przeplot 1         | przeplot 2         |
|---------------------|--------------------|--------------------|
| przepływ sterowania | {A} load $R_A, n$  | {A} load $R_A, n$  |
|                     | {A} add $R_A, 1$   | {A} add $R_A, 1$   |
|                     | {A} store $R_A, n$ | // $n = 0$         |
|                     | // $n = 1$         | {B} load $R_B, n$  |
|                     | {B} load $R_B, n$  | {B} add $R_B, 1$   |
|                     | {B} add $R_B, 1$   | {A} store $R_A, n$ |
|                     | {B} store $R_B, n$ | {B} store $R_B, n$ |
| wartość $n$         | 2                  | 1                  |

Współbieżność i synchronizacja procesów (16)

Systemy operacyjne

**Dane wejściowy i wyjściowe**

- Niech  $D_i^R \subseteq D_i$  oznacza zbiór danych czytanych przez proces  $P_i$ , a  $D_i^M \subseteq D_i$  oznacza zbiór danych modyfikowanych przez proces  $P_i$
- Dane wejściowe procesu  $P_i$  to takie dane współdzielone, które są czytane przez proces  $P_i$ , formalnie zatem jest to zbiór:
 
$$D_i^R \cap \bigcup_{1 \leq j \leq n, j \neq i} D_j$$
- Dane wyjściowe procesu  $P_i$  to takie dane współdzielone, które są modyfikowane przez proces  $P_i$ , formalnie zatem jest to zbiór:
 
$$D_i^M \cap \bigcup_{1 \leq j \leq n, j \neq i} D_j$$

Współbieżność i synchronizacja procesów (14)

Systemy operacyjne

**Przykład przeplotu instrukcji CISC**

|                     | przeplot 1  | przeplot 2  |
|---------------------|-------------|-------------|
| przepływ sterowania | {A} inc $n$ | {B} inc $n$ |
|                     | {B} inc $n$ | {A} inc $n$ |
| wartość $n$         | 2           | 2           |

Współbieżność i synchronizacja procesów (17)

Systemy operacyjne

**Przykład przetwarzania współbieżnego**

$n$ : integer := 0; /\* zmienna współdzielona \*/

|                 | wątek A                                         | wątek B                                         |
|-----------------|-------------------------------------------------|-------------------------------------------------|
| wysokopoziomowe | $n := n + 1$                                    | $n := n + 1$                                    |
| RISC            | load $R_A, n$<br>add $R_A, 1$<br>store $R_A, n$ | load $R_B, n$<br>add $R_B, 1$<br>store $R_B, n$ |
| CISC            | inc $n$                                         | inc $n$                                         |

Współbieżność i synchronizacja procesów (15)

Systemy operacyjne

**Istota synchronizacji**

- Celem synchronizacji jest kontrola przepływu sterowania pomiędzy procesami tak, żeby dopuszczalne stały się tylko przeploty instrukcji zgodne z intencją programisty.
- Synchronizacja na najniższym poziomie polega na wykonaniu specjalnych instrukcji, które powodują zatrzymanie postępu przetwarzania.
- Synchronizacja na wyższym poziomie polega na użyciu specjalnych konstrukcji programistycznych lub odpowiednich definicji struktur danych.

Współbieżność i synchronizacja procesów (18)

Systemy operacyjne



**Poprawność programów współbieżnych**

- Własność bezpieczeństwa (ang. safety) — w każdym stanie przetwarzania muszą być spełnione pewne warunki.
- Własność żywotności (ang. liveness) — w wyniku przetwarzania muszą w końcu zajść pewne warunki.

Współbieżność i synchronizacja procesów (19)

Systemy operacyjne



**Wzajemne wykluczanie — sformułowanie problemu**

- W systemie działa  $n$  procesów  $P_0, P_1, \dots, P_{n-1}$ .
- W programie każdego procesu znajduje się fragment kodu zwany sekcją krytyczną (ang. critical section).
- Sekcja krytyczna wykonywana jest w danej chwili przez co najwyżej jeden proces.

Współbieżność i synchronizacja procesów (22)

Systemy operacyjne



**Własność uczciwości programów współbieżnych**

- Uczciwość słaba — nieprzerwanie zgłaszane żądanie procesu będzie kiedyś obsłużone.
- Uczciwość mocna — nieskończenie wiele razy zgłaszane żądanie procesu będzie kiedyś obsłużone.
- Oczekiwanie liniowe — żądanie procesu będzie obsłużone po najwyżej jednokrotnym obsłużeniu żądań innych procesów.
- FIFO — żądania będą realizowane w kolejności zgłoszeń.

Współbieżność i synchronizacja procesów (20)

Systemy operacyjne



**Ogólna postać algorytmu wzajemnego wykluczania**

```

graph TD
    A[reszta (ang. remainder section)] --> B[sekcja wejściowa (ang. entry section)]
    B --> C[sekcja krytyczna (ang. critical section)]
    C --> D[sekcja wyjściowa (ang. exit section)]
    D --> A
  
```

Współbieżność i synchronizacja procesów (23)

Systemy operacyjne



**Klasyfikacja mechanizmów synchronizacji**

- Zapis lub odczyt współdzielonych danych
- Złożone operacje atomowe na współdzielonych danych (np. **test&set**, **exchange**)
- Mechanizmy wspierane przez system operacyjny
  - semafony
  - mechanizmy POSIX (zamki oraz zmienne warunkowe)
- Mechanizmy strukturalne (wspierane przez wysokopoziomowe języki programowania)
  - monitory
  - regiony krytyczne

Współbieżność i synchronizacja procesów (21)

Systemy operacyjne



**Poprawność rozwiązania problemu wzajemnego wykluczania**

- Wzajemne wykluczanie — warunek bezpieczeństwa,
- Postęp (ang. progress) — warunek żywotności z punktu widzenia systemu,
- Ograniczone czekanie (ang. lockout-freedom) — warunek żywotności z punktu widzenia procesu.

Współbieżność i synchronizacja procesów (24)

Systemy operacyjne

**Wzajemne wykluczanie 2 procesów — podejście 1**

Algorytm dla procesu  $P_i$  przy dwóch procesach  $P_i$  i  $P_j$

```

shared numer: Integer := i;

while numer ≠ i do
  nic;
sekcja krytycznai;
numer := j;
resztai;
  
```

sekcja wejściowa  
sekcja wyjściowa

Współbieżność i synchronizacja procesów (25)

Systemy operacyjne

**Wzajemne wykluczanie 2 procesów — podejście 4**

```

shared znacznik: array [0..1] of Boolean;
shared numer: Integer;

znacznik[i] := true;
numer := j;
while znacznik[j] and numer ≠ i do
  nic;
sekcja krytycznai;
znacznik[i] := false;
resztai;
  
```

sekcja wejściowa  
sekcja wyjściowa

Współbieżność i synchronizacja procesów (28)

Systemy operacyjne

**Wzajemne wykluczanie 2 procesów — podejście 2**

Algorytm dla procesu  $P_i$  przy dwóch procesach  $P_i$  i  $P_j$   
(dla uproszczenia prezentacji założono, że  $i = 0$ , a  $j = 1$ )

```

shared znacznik: array [0..1] of Boolean := false;

znacznik[i] := true;
while znacznik[j] do
  nic;
sekcja krytycznai;
znacznik[i] := false;
resztai;
  
```

sekcja wejściowa  
sekcja wyjściowa

Współbieżność i synchronizacja procesów (26)

Systemy operacyjne

**Wzajemne wykluczanie  $n$  procesów — algorytm piekarni <sup>(1)</sup>**

Algorytm dla procesu  $P_i$  przy  $n$  procesach ( $i = 0 \dots n-1$ )

```

shared wybieranie: array [0..n-1] of Boolean := false;
shared numer: array [0..n-1] of Integer := 0;

local k: Integer;

wybieranie[i] := true;
numer[i] := max(numer[0], numer[1], ...) + 1;
wybieranie[i] := false;
  
```

drzwi

Współbieżność i synchronizacja procesów (29)

Systemy operacyjne

**Wzajemne wykluczanie 2 procesów — podejście 3**

```

shared znacznik: array [0..1] of Boolean := false;

znacznik[i] := true;
while znacznik[j] do
begin
  znacznik[i] := false;
  znacznik[i] := true;
end;
sekcja krytycznai;
znacznik[i] := false;
resztai;
  
```

sekcja wejściowa  
sekcja wyjściowa

Współbieżność i synchronizacja procesów (27)

Systemy operacyjne

**Wzajemne wykluczanie  $n$  procesów — algorytm piekarni <sup>(2)</sup>**

```

1) for k := 0 to n-1 do begin
2)   if k ≠ i then begin
3)     while wybieranie[k] do nic;
4)     while numer[k] ≠ 0 and
5)       (numer[k], k) < (numer[i], i) do
6)       nic;
7)   end;
8) end;
9) sekcja krytycznai;
10) numer[i] := 0;
11) resztai;
  
```

sekcja wejściowa  
sekcja wyjściowa

Współbieżność i synchronizacja procesów (30)

Systemy operacyjne



**Operacja test&set**

```

function test&set(var l: Boolean): Boolean;
begin
    test&set := l;
    l := true;
end;
  
```

Współbieżność i synchronizacja procesów (31)

Systemy operacyjne



**Wzajemne wykluczanie z użyciem instrukcji exchange**

```

shared zamek: Boolean;
local klucz: Boolean;

klucz := true;
repeat exchange(zamek, klucz);
until klucz = false;
sekcja krytyczna;
zamek := false;      ← sekcja wyjściowa
reszta;
  
```

sekcja wejściowa

Współbieżność i synchronizacja procesów (34)

Systemy operacyjne



**Operacja exchange**

```

procedure exchange(var a, b: Boolean);
var tmp: Boolean;
begin
    tmp := a;
    a := b;
    b := tmp;
end;
  
```

Współbieżność i synchronizacja procesów (32)

Systemy operacyjne



**Wzajemne wykluczanie z użyciem instrukcji test&set**

```

shared zamek: Boolean;

while test&set(zamek) do
    nic;
sekcja krytyczna;
zamek := false;      ← sekcja wyjściowa
reszta;
  
```

sekcja wejściowa

Współbieżność i synchronizacja procesów (33)