

# PHP – PHP Hypertext Preprocessor

Igor Wojnicki

Katedra Automatyki  
Akademia Górniczo-Hutnicza w Krakowie

13 maja 2012

\$Id: php1.tex,v 1.1 2011/11/19 19:14:25 wojnicki Exp wojnicki \$

- 1 PHP
  - Współbieżność
  - Ciastka i sesje

I. Wojnicki, Tech.Inter.

# Spis Treści

- 1 PHP
  - Współbieżność
  - Ciastka i sesje

I. Wojnicki, Tech.Inter.

# Wyścig

- Problem: Wyścig – dwa procesy (lub więcej) żądają dostępu do tego samego dzielonego zasobu (pamięć dzielona, plik, obiekt w bazie danych).
- Tylko jeden może uzyskać dostęp – kto pierwszy ten lepszy.
- Przykłady:
  - dwa procesy zapisują ten sam plik,
  - atomiczność przy złożonych operacjach odczytu/zapisu (bazy danych).

# Wzajemne wykluczenie

## Wzajemne wykluczenie

Mutual Exclusion – jeżeli jeden proces używa dzielonego zasobu, żaden inny proces nie może go używać.

## Sekcja krytyczna

Critical Region/Section: część kodu programu realizująca dostęp do dzielonego zasobu.

# Problemy

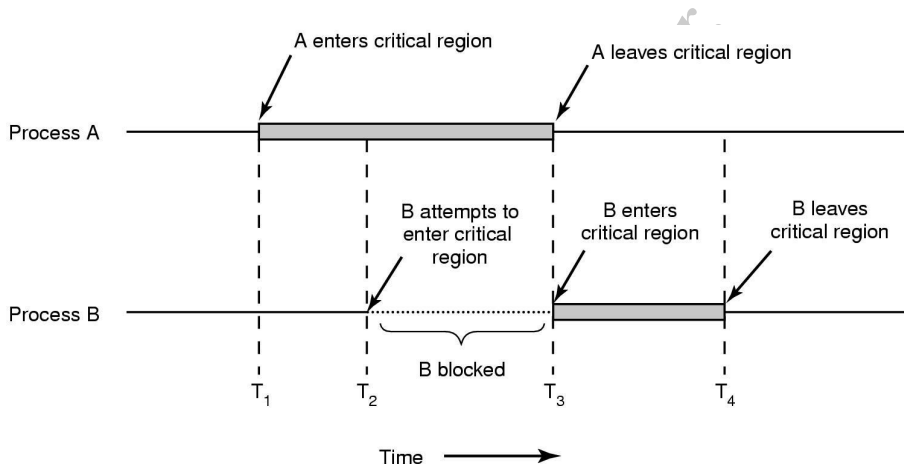
- Zagłódzenie
- Zakleszczenie.

I. Wojnicki, Tech.Inter.

# Warunki dla wzajemnego wykluczenia

- 1 Tylko jeden proces może realizować sekcję krytyczną.
- 2 Szybkość sprzętu nie może być brana pod uwagę.
- 3 Proces poza sekcją krytyczną nie może zablokować innego procesu.
- 4 Proces nie może w nieskończoność być blokowany przed wejściem do sekcji krytycznej.

# Sekcja krytyczna, przykład





# Jak zaimplementować wzajemne wykluczenie w PHP?

- mutex (semafor binarny)
- semafor (liczący)
- semafor na pliku
- let the others do – np. transakcje w bazach danych

I. Wojnicki, Tech.Inter.

# Mutex

- Semafor binarny.
- Stany:
  - 1 – odblokowany,
  - 0 – zablokowany.
- Operacje (atomiczne!):
  - w dół –  
`if (stan==1) {stan=0; kontynuuj;} else czekaj`
  - w górę – `stan=1; obudź czekającego`

# Semafor

- Stan: licznik.
- Operacje (atomiczne!):
  - w dół – `if (stan>0) {stan--; kontynuuj;} else czekaj`
  - w górę – `stan++; obudź czekającego`

I. Wojnicki, Tech.Inter.

# Przykład: semafor

## semafor, sekcja krytyczna

```
define(KLUCZ,123456);
$sem=sem_get(KLUCZ);
echo 'Wejście do sekcji krytycznej<br>';
ob_flush(); flush();
$start=time();
sem_acquire($sem);
echo 'Jestem w sekcji krytycznej<br>';
ob_flush(); flush();
sleep(5);
sem_release($sem);
echo 'Wyjście z sekcji krytycznej<br>';
echo 'Czas wykoania: ' .(time()-$start).'<br>';
```

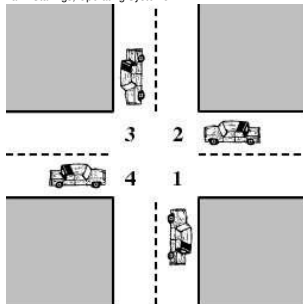
# File Locking

Kontrola dostępu do plików; semafor: flock()

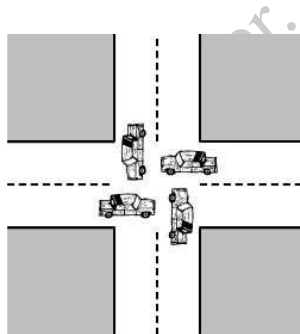
```
echo 'Wejście do sekcji krytycznej<br>';
ob_flush(); flush();
$start=time();
$fp = fopen('semafor', 'r+');
if (flock($fp, LOCK_EX)) {
    echo 'Jestem w sekcji krytycznej<br>';
    ob_flush(); flush();
    sleep(5);
    flock($fp, LOCK_UN);
    echo 'Wyjście z sekcji krytycznej<br>';
    echo 'Czas wykoania: ' .(time()-$start).' <br>';
} else { echo 'Problemy z blokada...<br>'; }
ob_flush(); flush(); fclose($fp);
```

# Zakleszczenie, z życia

za W.Stallings, Operating Systems



(a) Deadlock possible

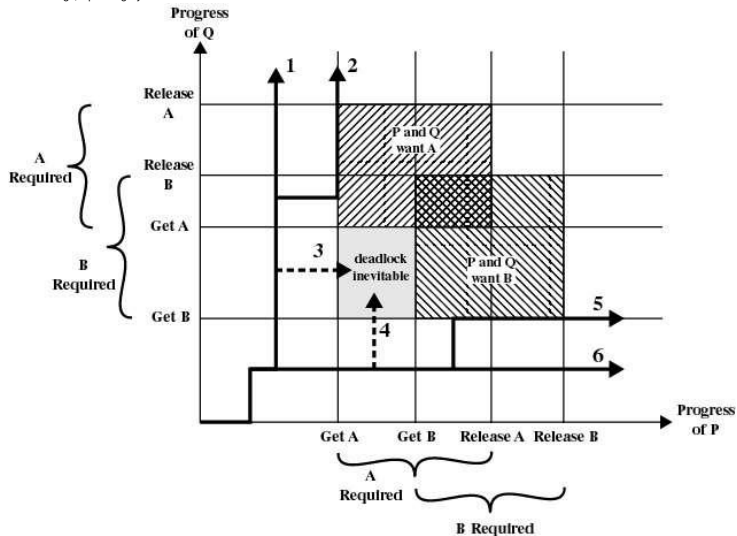


(b) Deadlock

Figure 6.1 Illustration of Deadlock

# Zakleszczenie, z informatyki

za W.Stallings, Operating Systems



# Zakleszczenie

Zbiór procesów jest zakleszczony jeżeli każdy z nich czeka na zdarzenie, które tylko inny proces ze zbioru może spowodować.

- Zwykle zdarzeniem jest zwolnienie zasobu.
- Żaden z procesów nie może:
  - kontynuować wykonania,
  - zwolnić zasobu,
  - zostać odblokowanym.



# Zakleszczenie, rozwiązanie

- Dostęp do najwyżej jednego zasobu na raz.
- Przydział zasobów w określonej (takiej samej) kolejności dla wszystkich procesów.

I. Wojnicki, Tech.Inter.

# Spis Treści

- 1 PHP
  - Współbieżność
  - Ciastka i sesje

I. Wojnicki, Tech.Inter.

# Stan

- WWW – architektura klient-serwer.
- Bezstanowa.
- Identyfikacja klienta → stanowa:
  - ciastka (cookies),
  - sesja.

I. Wojnicki, Tech.Inter.

# Cookies

- Permanentne przechowywanie informacji po stronie przeglądarki.
- Przesyłane w nagłówku HTTP.

I. Wojnicki, Tech.Inter.

# Obsługa cookies

- `$_COOKIE['nazwa_ciastka']`
- `setcookie()`
  - można użyć jedynie zanim skrypt wygeneruje wyjście

I. Wojnicki Tech.Inter.

# setcookie()

```
setcookie(string $name [, string $value
    [, int $expire = 0 [, string $path
    [, string $domain [, bool $secure = false
    [, bool $httponly = false ]]]]] ) |
```

- Po wywołaniu, wartość przesyłana jest do przeglądarki.
- Przy następnym odczycie jest dostępna w `$_COOKIE` albo `$_REQUEST`.
- `name` – nazwa
- `value` – wartość (opcjonalnie, domyślnie pusty łańcuch znaków)
- `expire` – data przydatności do spożycia:  
`time() + liczba_sekund`, domyślnie 0 – koniec sesji przeglądarki
- `path` – ścieżka w jakiej będzie dostępna (/ w całej domenie, albo dla konkretnej ścieżki na serwerze)
- Unieważnienie: ustawienie odpowiedniej wartości `expire`

# Cookies, dobre rady

- `setcookie()` zwraca `FALSE` jeżeli przesłanie danych w nagłówku HTTP się niepowiodło (wygenerowane wyjście, przed wywołaniem funkcji).
- `htmlspecialchars()` oraz `htmlspecialchars_decode()` – do konwersji znaków specjalnych m.in. `<`, `>`, apostrof, cudzysłów.

# Sesja

- Implementacja stanu aplikacji.
- Przechowywanie danych pomiędzy wywołaniami skryptów w `$_SESSION`.

I. Wojnicki, Tech.Inter.



# session\_start ()

- Tworzy sesje, lub przywraca rozpoczętą.
- Zwraca FALSE jeżeli operacja się nie udała.
- Dane zapisane/odczytane z `$_SESSION` będą dostępne dla skryptów.
- Identyfikator sesji przekazywany przez ciastko (albo GET/POST – niezalecane).

I. Wojnicki, Tech-Inter.

# Unieważnienie danych w sesji

- `unset($_SESSION['moje_dane'])`
- Dane przechowywane w sesji są zapisywane po stronie serwera.
- Fizyczny zapis danych następuje przy zakończeniu działania skryptu (można wymusić wcześniej).
- Tylko jeden proces może zapisywać/odczytywać dane konkretnej sesji na raz.