PROCESY I WĄTKI

WYKŁAD II

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

PROCESY

Ten wykład rozpoczyna segment kursu, który obejmuje procesy, wątki i synchronizację

- Tematy te są chyba najważniejsze w tematyce systemów operacyjnych
- Na pewno będą objęte testem

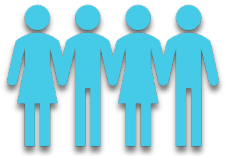
Dzisiejsze tematy to: procesy i zarządzanie nimi

- Co to są procesy?
- Jak reprezentowane są procesy w systemie operacyjnym?
- Jakie są możliwe stany wykonania procesu?
- Jak proces przechodzi z jednego stanu do drugiego?

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

OPIS ELEMENTÓW SYSTEM PRZED ROZWAŻENIEM PROCESU

Użytkownicy (wielu) mają konta w systemie



Użytkownicy uruchamiają programy

Przypadek 1: Wielu użytkowników może uruchomić ten sam program

Przypadek 2: Jeden użytkownik może uruchomić wiele instancji tego samego programu

Co to jest proces?

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

PROCESS (DEFINICJA)

Proces jest abstrakcją wykonywanego programu

- To inaczej jednostka wykonania (execution)
- Jest to jednostka podlegająca planowaniu
- Jest to dynamiczny kontekst wykonania programu

Proces nazywany jest czasem **zadaniem** lub **pracą** lub procesem sekwencyjnym

Choć w systemach wielozadaniowych pozornie wykonuje się równolegle wiele procesów, to są obsługiwane sekwencyjnie.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

ANALOGIA – ROBOT UCZESTNICZĄCY W WYKŁADACH Z OS

Program: kroki udziału w wykładzie

- Krok 1: przejdź do Sali wykładowej
- Krok 2: znajdź miejsce
- Krok 3: słuchaj (lub śpij)

Proces: udział w wykładzie

- Akcja – to co się dzieje
- Wszyscy jesteście w trakcie procesu

TYPY PROCESÓW

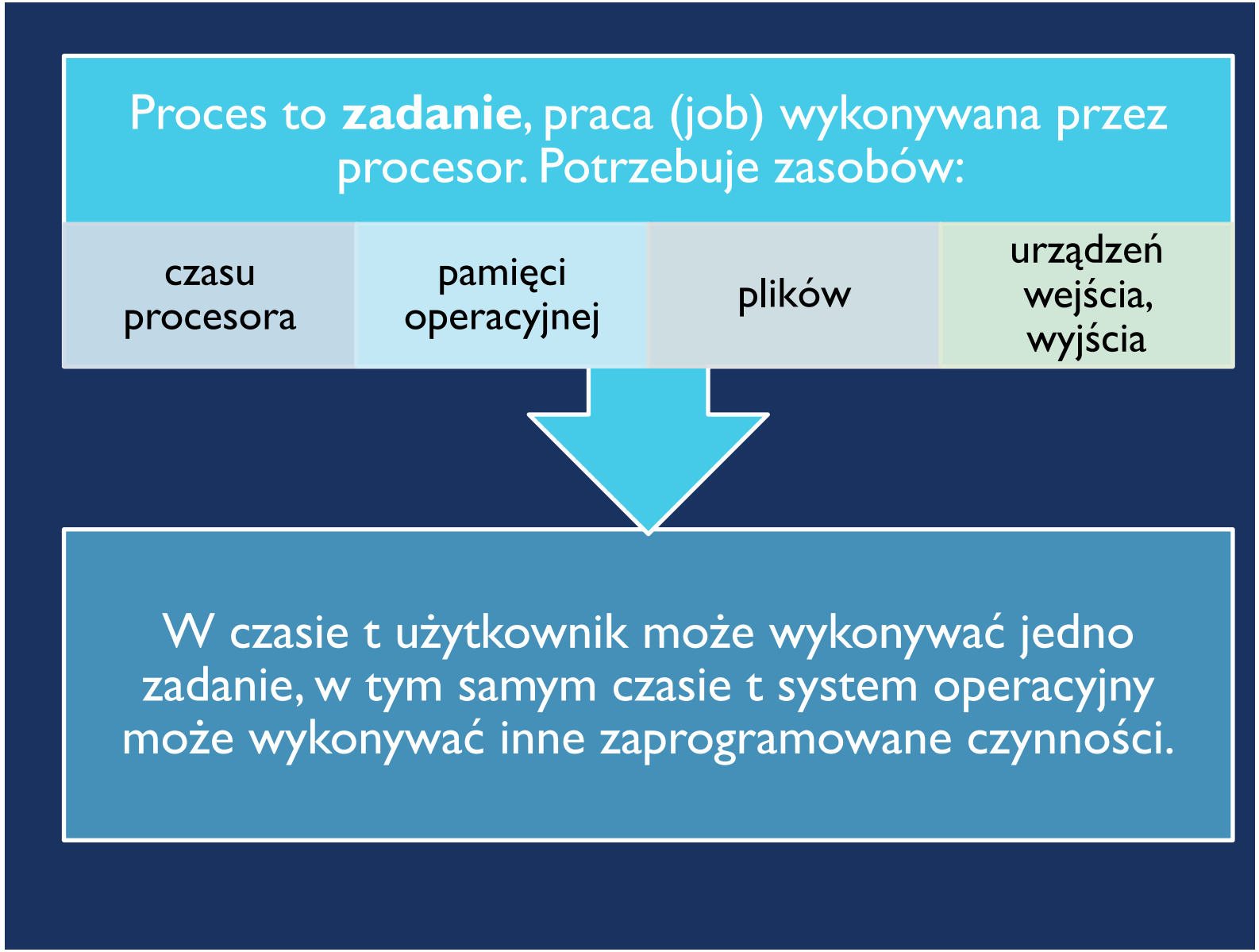
Procesor wykonuje wiele programów: systemu i użytkowych. Działania te nazywany **procesami**.

Procesy systemu operacyjnego to wykonujące się programy systemowe.

Procesy użytkowe są uruchomionymi programami napisanymi przez użytkownika (uruchomionych przez użytkownika).

Wszystkie procesy mogą działać **współbieżnie**, czyli zdolności obliczeniowe procesora są dzielone pomiędzy poszczególne procesy.

KONCEPCJA PROCESU



AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

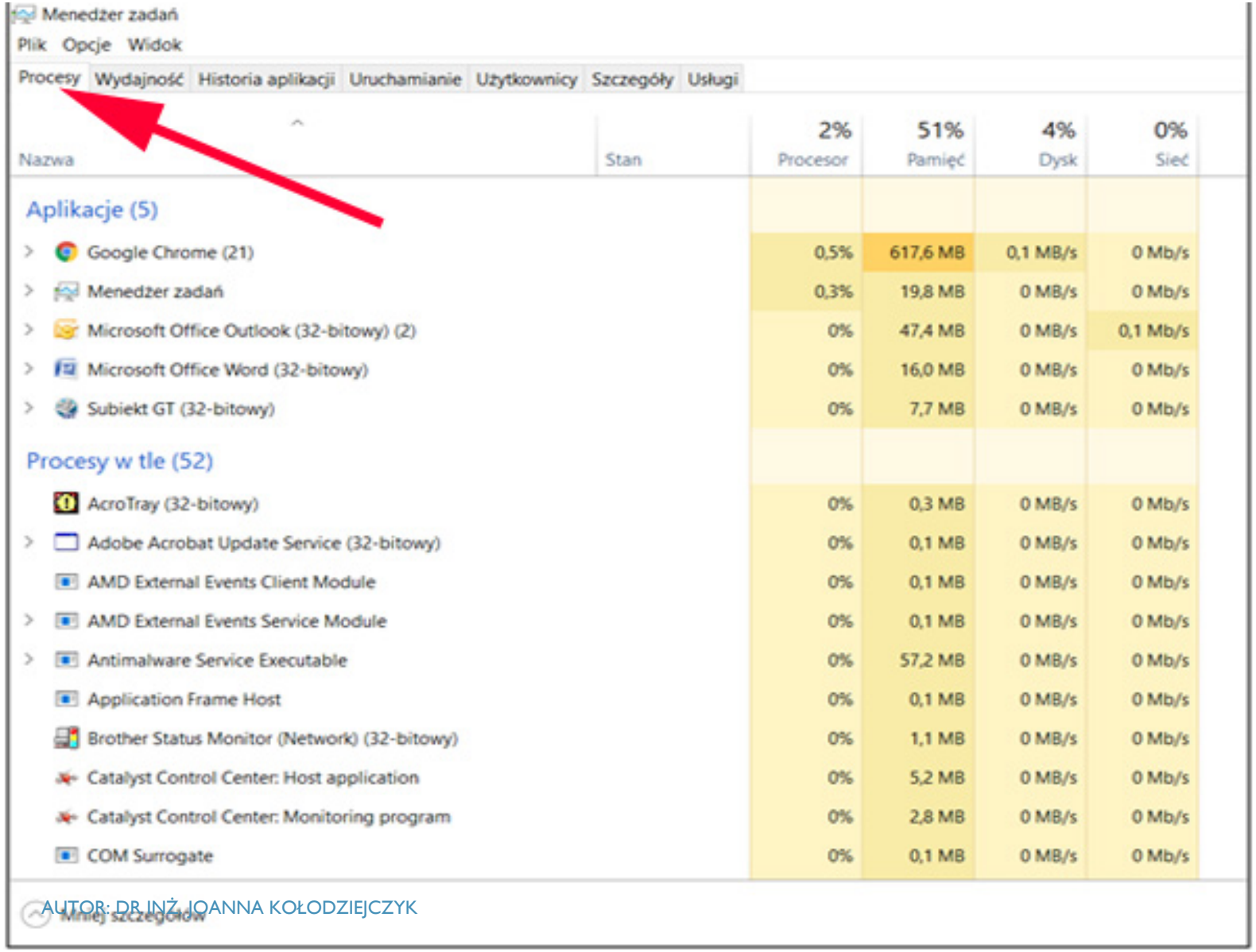
MAC OS PRZYKŁAD

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	PID	User
WindowServer	8.3	6:21:59.24	4	29	187	_windowserver
hidd	4.5	36:01.16	6	0	115	_hidd
kernel_task	3.8	5:34:15.29	250	185	0	root
screencapture	2.6	0.38	2	0	33124	ding
Activity Monitor	2.6	1:59:58.29	6	2	617	ding
Acrobat	2.0	12:46.87	20	90	31159	ding
Microsoft PowerPoint	1.2	2:06:48.28	13	25	2969	ding
distnoted	1.1	2:04:40.47	11	0	283	ding
sysmond	0.8	1:35:33.77	3	1	274	root
distnoted	0.5	1:00:52.69	8	0	118	_distnote
Google Chrome Helper	0.4	13:55.32	18	8	23466	ding
CLion	0.4	1:45:20.67	43	46	2567	ding
Dropbox	0.3	53:34.30	191	4	13921	ding
com.apple.AmbientDi...	0.3	8:00.60	5	0	220	root
iTerm2	0.3	30:50.54	7	5	2610	ding
Adobe Reader and A...	0.2	18.24	5	2	32864	ding
Google Chrome Helper	0.2	10:05.63	19	2	23697	ding
AdobeCrashDaemon	0.2	22:25.51	1	1	469	ding
launchservicesd	0.2	15:29.24	3	0	98	root
Google Chrome	0.1	4:11:59.24	41	0	13353	ding

Thanks for flying Vim

```
diyuan@ug132:~$ ps -e
  PID TTY          TIME CMD
    1 ?        00:00:04 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 migration/0
    4 ?        00:00:00 ksoftirqd/0
    5 ?        00:00:00 watchdog/0
    6 ?        00:00:00 migration/1
    7 ?        00:00:00 ksoftirqd/1
    8 ?        00:00:00 watchdog/1
    9 ?        00:00:00 migration/2
   10 ?        00:00:00 ksoftirqd/2
   11 ?        00:00:00 watchdog/2
   12 ?        00:00:00 migration/3
   13 ?        00:00:00 ksoftirqd/3
   14 ?        00:00:00 watchdog/3
   15 ?        00:00:00 events/0
   16 ?        00:00:00 events/1
   17 ?        00:00:03 events/2
   18 ?        00:00:00 events/3
   19 ?        00:00:00 cpuset
   20 ?        00:00:00 khelper
   21 ?        00:00:00 netns
   22 ?        00:00:00 async/mgr
```

LINUX PRZYKŁAD



WINDOWS 10 PRZYKŁAD

Proces to program w trakcie wykonania

Jest to pojedyncza wykonująca się instancja programu

Jest odseparowany od innych instancji

Może rozpocząć odpalenie („launch”) innych procesów

Może zostać uruchomiony przez inny proces

CECHY PROCESÓW

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

tworzenie

usuwanie

planowanie porządku wykonywania procesów

synchronizowanie

komunikację

pokonywanie blokad

ROLA SYSTEMU OPERACYJNEGO W ZARZĄDZANIU PROCESAMI

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

PROCES JEST SEKWENCYJNY

Wykonanie procesu odbywa się w sposób sekwencyjny: w danej chwili na zamówienie procesu może być wykonany co najwyżej jeden rozkaz kodu programu.

Program jako plik nie jest procesem, ponieważ jest pasywny - jest przechowywany na dysku.

Proces jest aktywny, ma licznik rozkazów określający następny rozkaz do wykonania.

Z jednym programem może być związana pewna liczba procesów. Zazwyczaj wykonujący się program (proces) tworzy wiele nowych procesów.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

MOŻLIWE STANY PROCESU

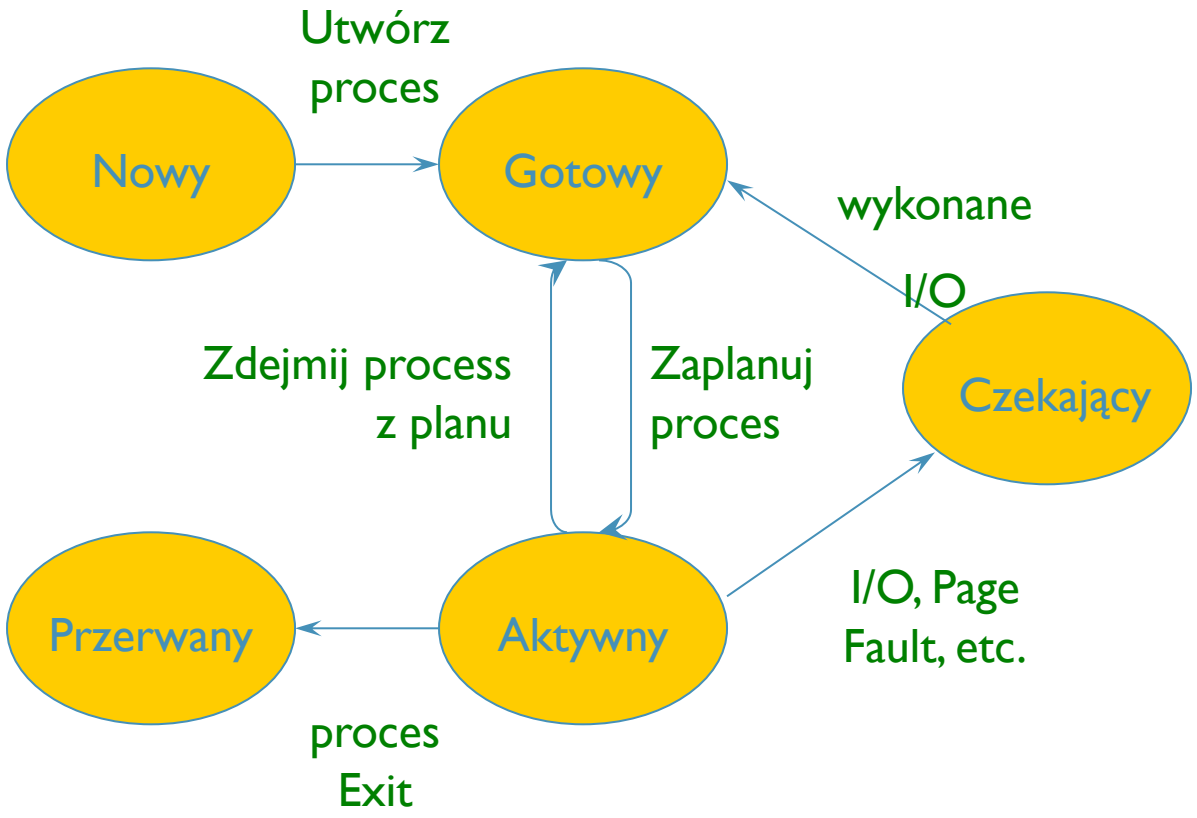
AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

Proces zmienia swój stan i w danej chwili może znajdować się w jednym ze stanów:

1. **aktywny**: wykonuje instrukcje, - jest to proces, który wykorzystuje procesor
2. **czekający**: czeka na wystąpienie jakiegoś zdarzenia (np.. operacji wejścia-wyjścia),
3. **gotowy**: czeka na przydział procesora. - gotowy do wykonania, ale inny proces zajmuje CPU

Tylko jeden proces może być **aktywny**, ale wiele może być w oczekiwaniu bądź w gotowości.

SCHEMAT ZMIANY STANU PROCESU



PYTANIA

- Jak myślisz, w jakim stanie proces spędza najwięcej czasu?
- Dla maszyny jednoprocesorowej ile procesów może być w stanie uruchomionym?
- Korzyść procesora wielordzeniowego?
- Ile procesów może być jednocześnie uruchomionych?

TWORZENIE PROCESU

Proste systemy (kontroler kuchenki mikrofalowej) wszystkie procesu inicjowane są często od razu.

Systemy ogólnego przeznaczenia muszą manipulować procesami.

Zdarzenia powodujące tworzenie procesów:

- Inicjalizacja systemu.
- Wykonanie wywołania systemowego tworzenia procesu przez uruchomiony proces.
- Żądanie użytkownika, aby utworzyć nowy proces.
- Rozpoczęcie zadania wsadowego.

ZAKOŃCZENIE PROCESU

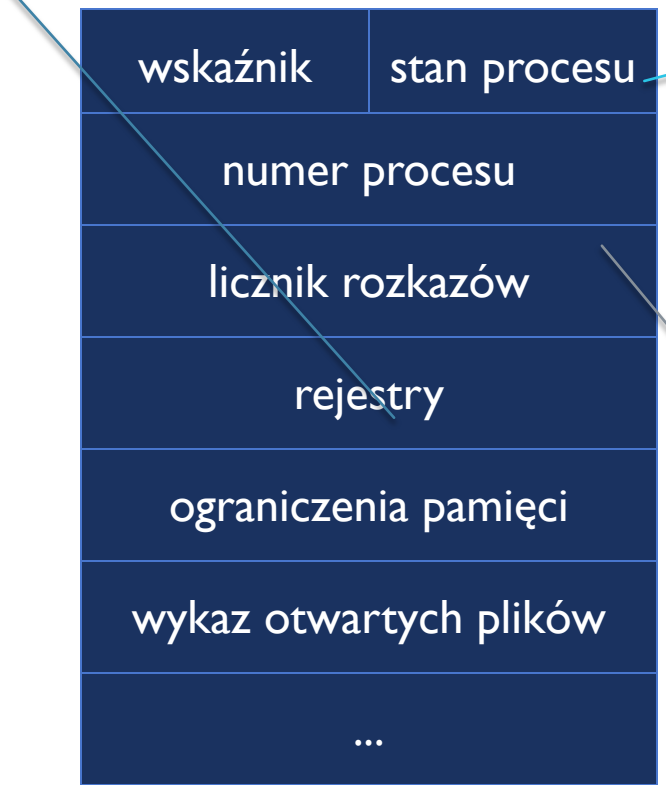
- Przyczyny zakończenia procesu
  - Normalne wyjście (dobrowolne).
  - Błąd wyjścia (dobrowolny).
  - Błąd krytyczny (mimowolny).
  - Zabity przez inny proces (mimowolny).

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

BLOK KONTROLNY PROCESU

Blok danych zawierających informacje o procesie.

Zależy od architektury. Podczas przerwania stan rejestrów musi zostać przechowany, by potem poprawnie kontynuować proces



1. nowy
2. gotowy
3. aktywny
4. czekający
5. wstrzymany

wskazuje adres następnej instrukcji do wykonania w procesie

INNE ELEMENTY BLOKU PROCESU

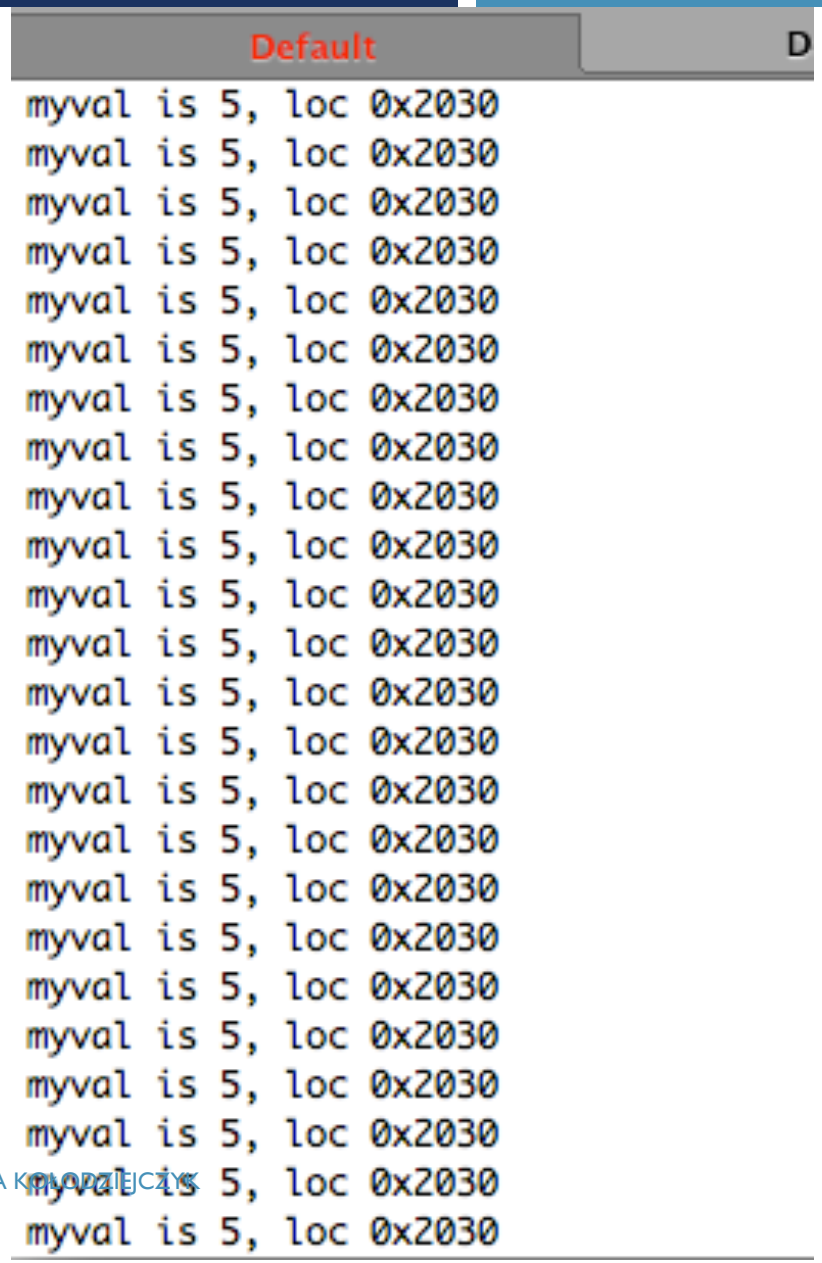
Informacje o harmonogramowaniu procesora	• priorytet procesu;
Informacje dotyczące zarządzania pamięcią	• informacje o bazie / limicie, wirtualne -> mapowanie fizyczne itp
Informacje identyfikujące	• limity czasowe, numer procesu; właściciel
Informacje o stanie we / wy	• lista urządzeń I / O przydzielonych do procesu;
Przestrzeń adresowa	• przestrzeń pamięci widoczna dla jednego procesu

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

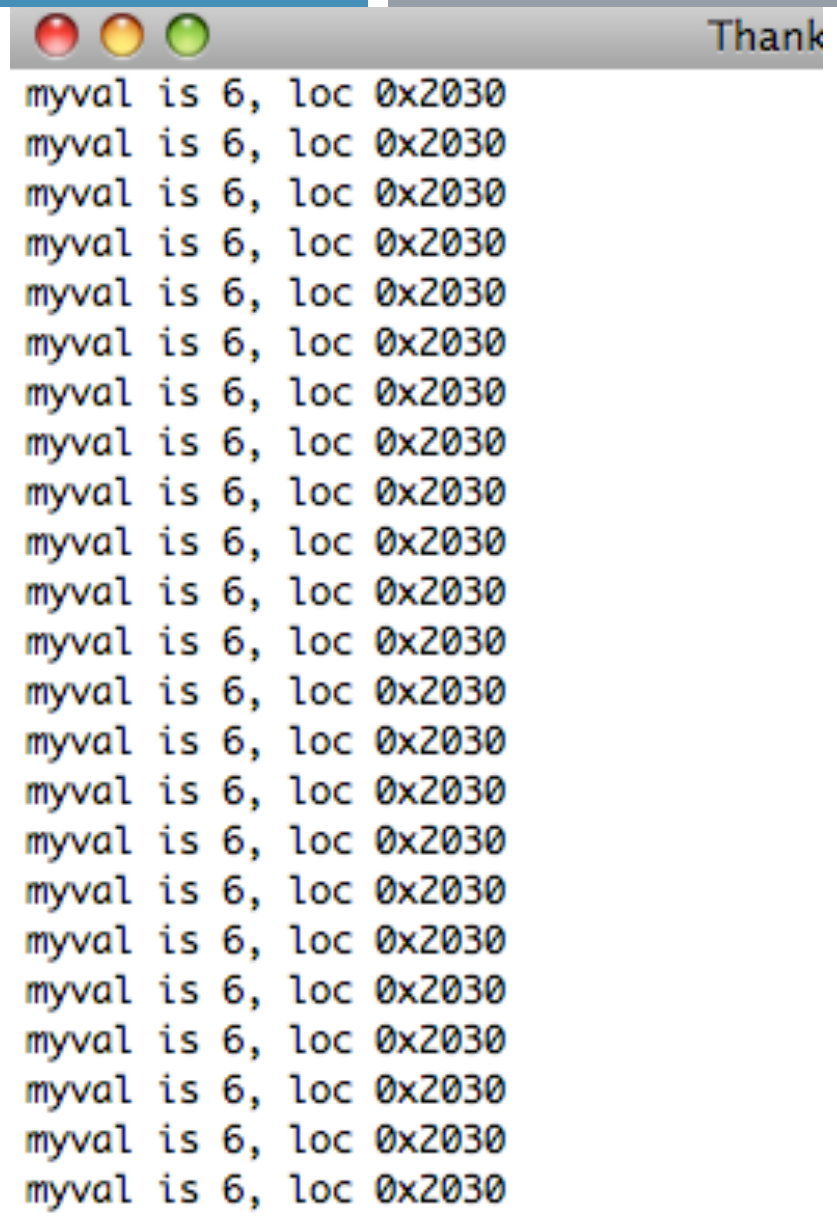
A CO JEŻELI?

```
int myval;
int main(int argc, char *argv[])
{
  myval = atoi(argv[1]);
  while (1)
   printf("myval is %d, loc 0x%lx\n", myval, (long) &myval);
}
```

- Teraz jednocześnie uruchom dwa razy program z dwoma różnymi parametrami wejściowymi:
  1. Myval 5
  2. Myval 6
- Jakie będą wyniki?



```
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
myval is 5, loc 0x2030
```



```
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
myval is 6, loc 0x2030
```

ANALIZA PRZYKŁADU

- Adres zmiennej **myval** był zawsze taki sam, ale wartości były różne.
- Wniosek:
  1. Programy nie widzą się
  2. Ale myślą, że używają tego samego adresu
- Konkluzja:
  - adresy nie są „pamięcią fizyczną”
- W jaki sposób taki stan zrealizować?
  - Mapowanie pamięci
- Jakie są korzyści?

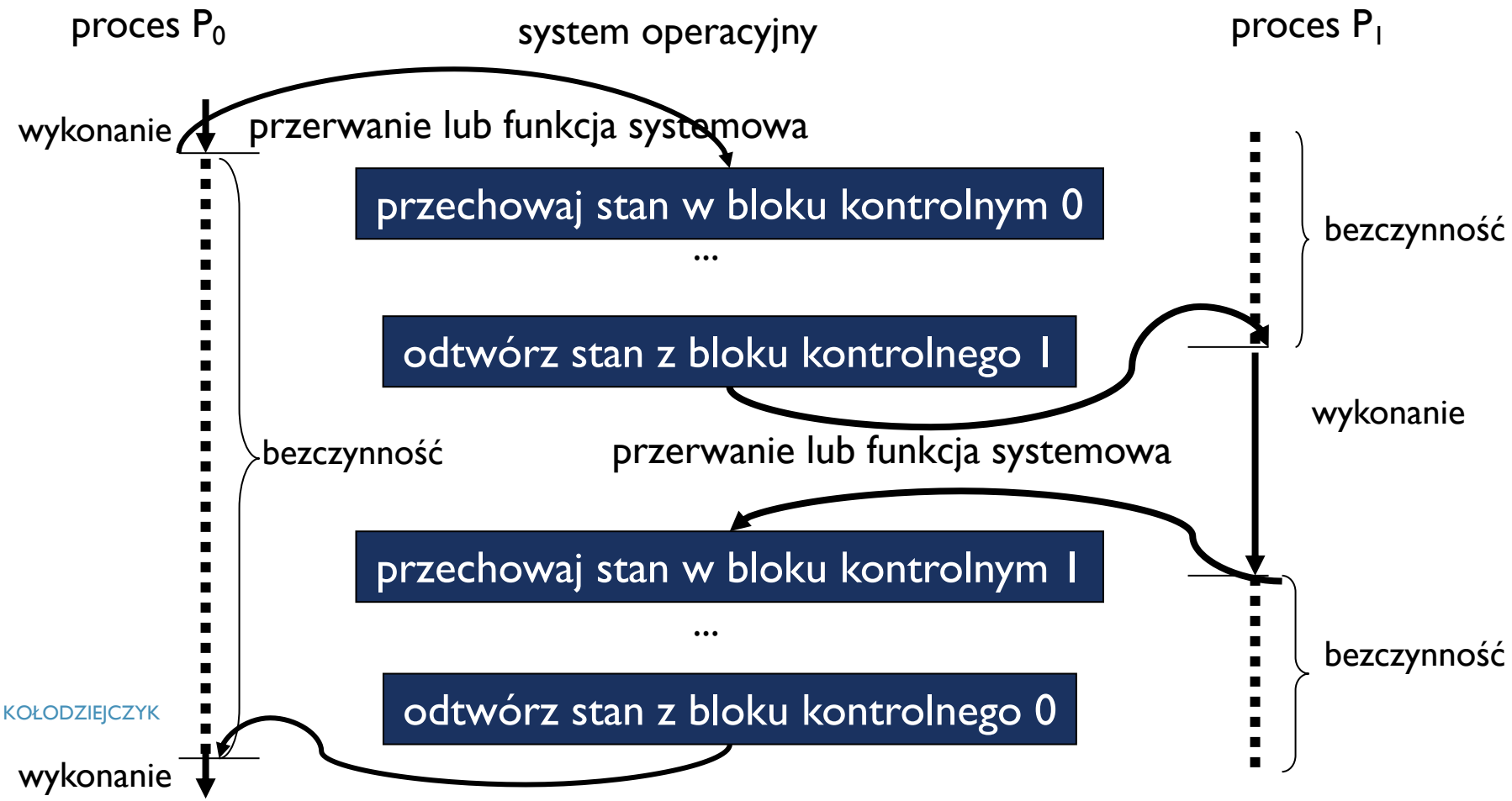
PROCESY WSPÓŁBIEŻNE

Procesy mogą działać **współbieżnie** – procesor dzieli swoją moc obliczeniową pomiędzy wiele procesów.

Zalety współbieżności:

1. podział zasobów fizycznych,
2. podział zasobów logicznych (współdzielenie informacji: plików),
3. przyspieszenie obliczeń (zadanie dzielimy na podzadania i wykonujemy równolegle, duże przyspieszenie, zwłaszcza gdy istnieje wiele elementów przetwarzających),
4. modularność,
5. wygoda: działania współbieżne, nawet dla jednego użytkownika (wykonywane wielu działań: drukowanie, edycja pliku, odtwarzanie dźwięku).

REALIZACJA PROCESÓW WSPÓŁBIEŻNIE



TWORZENIE PROCESU – WIĘCEJ SZCZEGÓŁÓW

- Proces **macierzysty** – proces tworzący nowe procesy (za pomocą specjalnej funkcji systemowej). Pamięta identyfikatory swoich potomków.
- Proces **potomny** – proces powstały na żądanie innego procesu.
- Zasoby (czas procesora, pamięć, pliki, urządzenia wejścia-wyjścia) mogą dla procesu być przydzielane:
  - przez system operacyjny, każdy proces walczy o ogólne zasoby z innymi procesami na równi (np. Unix),
  - ograniczone do podzbioru zasobów procesu macierzystego.
- Proces macierzysty może rozdzielać swoje zasoby pomiędzy potomków, lub pozwalać potomkom współdzielić zasoby. Ograniczenie zasobów zapobiega rozmnażaniu się procesów.
- Wytworzenie nowego procesu może prowadzić do dwóch scenariuszy:
  1. proces macierzysty kontynuuje współbieżność z potomkami,
  2. proces macierzysty czeka, aż wszystkie procesy potomne zakończą pracę.

ZAKOŃCZENIE PROCESU – WIĘCEJ SZCZEGÓŁÓW

- Proces po ostatniej instrukcji prosi SO o usunięcie.
- Dane po zakończeniu procesu potomnego, jeżeli proces macierzysty tego żąda można przekazać do procesu macierzystego.
- Zakończenie podprocesu przez proces macierzysty może nastąpić z powodu:
  - potomek nadużył przydzielonego zasobu,
  - wykonywane przez potomka zadanie stało się zbędne.
- Zakończenie **kaskadowe**: zamknięcie wszystkich procesów potomnych zamykanego procesu.

Procesy **niezależne**:

- Na jego stan nie wpływa żaden inny proces, ani on nie wpływa na inne procesy.
- Nie dzieli danych z innymi procesami.
- Jego działanie jest deterministyczne: wynik zależy wyłącznie od stanu wejściowego.
- Jego działanie daje się powielać, przy tych samych danych, zawsze otrzyma się taki sam wynik.
- Jego działanie może być wstrzymane i wznawiane bez szkodliwego oddziaływania.

Procesy **współpracujące**:

- Stan dzielony z innymi procesami,
- Nie da się z góry określić działania, ponieważ zależy od względnej kolejności jego wykonania,
- Wynik niedeterministyczny: przy tych samych danych wejściowych wynik może być różny.

DWA TYPY PROCESÓW

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

PRZERWA

WĄTKI - CECHY

Nazywane lekkimi procesami. Są to współbieżne zadania w obrębie jednego procesu.

W SO wielowątkowym proces zawiera co najmniej 1 wątek sterowania. Wątki konkurują o czas procesora. Pozostałe zasoby (przestrzeń adresowa) przydziale są dla całego procesu.

W systemach bez wątków proces i wątek jest utożsamiany.

Obsługa współbieżności wątków może być realizowana na różne sposoby. Są one podobne do obsługi procesów.

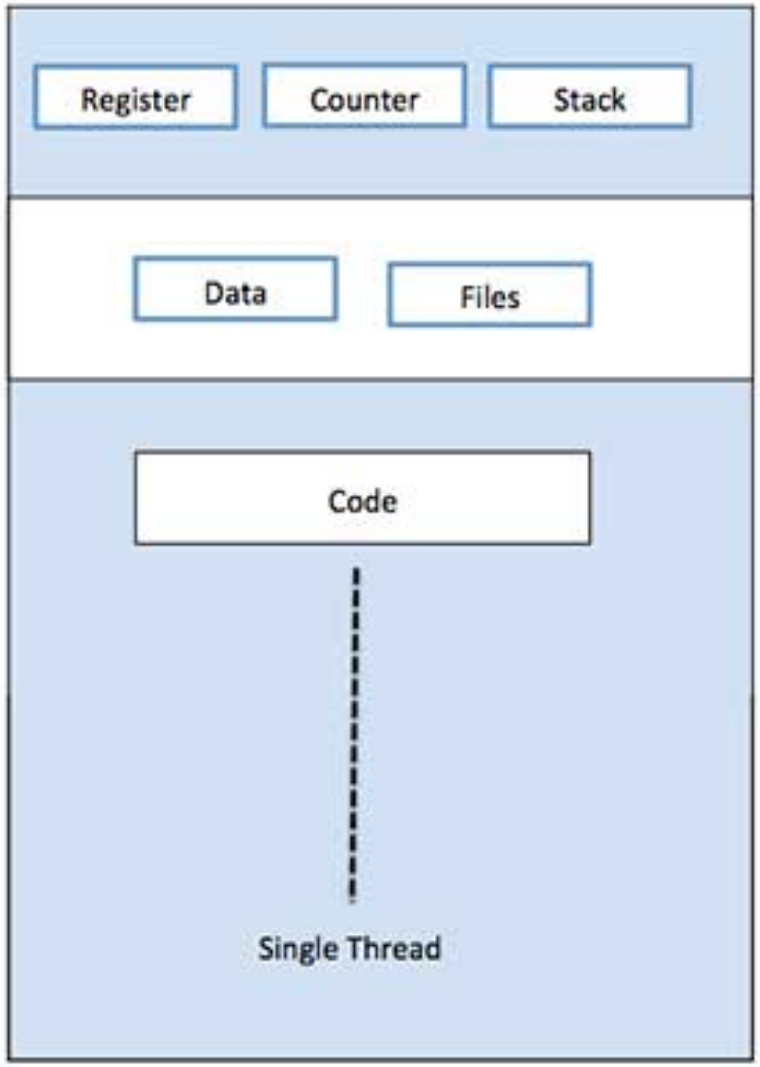
RÓŻNICE POMIĘDZY PROCESAMI I WĄTKAMI

LP	Proces	Wątek
1	Proces jest ciężki lub wymaga dużej ilości zasobów.	Wątek jest lekki, zużywa mniej zasobów niż proces.
2	Przełączanie procesów wymaga interakcji z systemem operacyjnym.	Przełączanie wątków nie wymaga interakcji z systemem operacyjnym.
3	W wielu środowiskach każdy proces wykonuje ten sam kod, ale ma własną pamięć i zasoby plików.	Wszystkie wątki mogą współdzielić ten sam zestaw otwartych plików, procesy potomne.
4	Jeśli jeden proces jest zablokowany, żaden inny proces nie może zostać wykonany, dopóki pierwszy proces nie zostanie odblokowany.	Jeden wątek jest zablokowany i czeka, drugi wątek w tym samym procesie może zostać uruchomiony.
5	Wiele procesów bez użycia wątków wykorzystuje więcej zasobów.	Wiele procesów z wątkami zużywa mniej zasobów.
6	W wielu procesach każdy proces działa niezależnie od innych.	Jeden wątek może odczytywać, zapisywać lub zmieniać dane innego wątku.

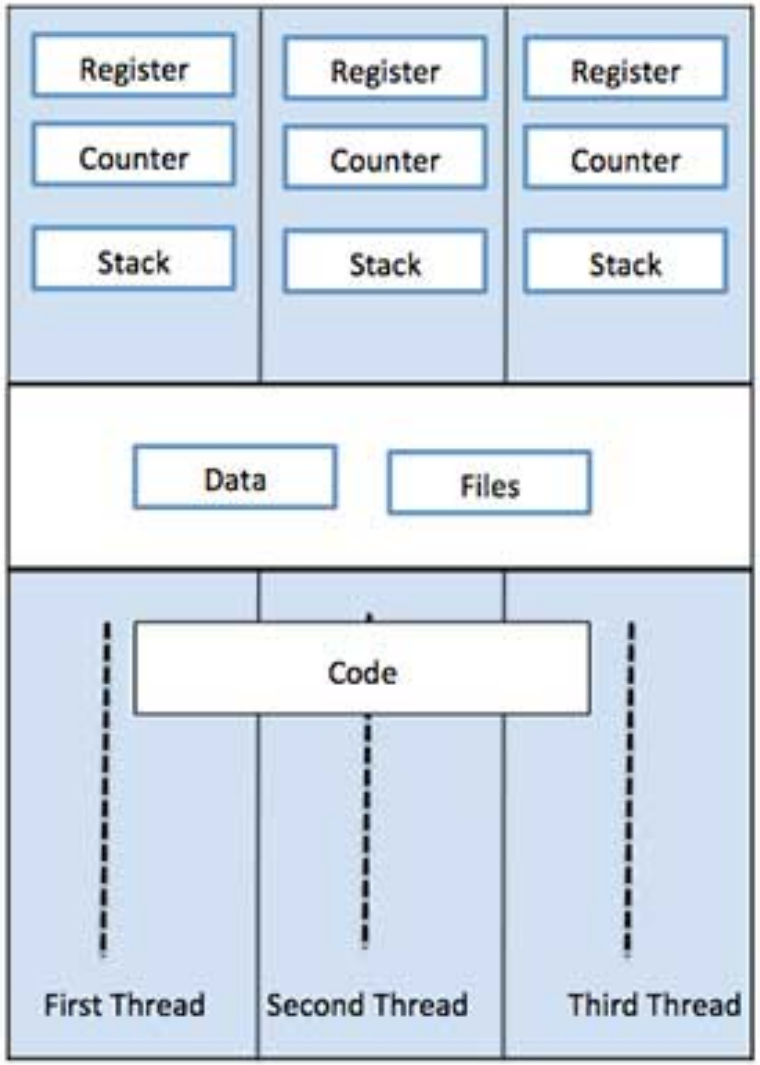
CECHY WĄTKÓW

- Każdy wątek należy do dokładnie jednego procesu i żaden wątek nie może istnieć poza procesem.
- Każdy wątek reprezentuje oddzielny przepływ sterowania.
- Wątki zostały z powodzeniem zastosowane w implementacji serwerów sieciowych i serwera WWW.
- Stanowią także podstawę do równoległego wykonywania aplikacji na maszynach wieloprocesorowych z pamięcią współdzieloną.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK



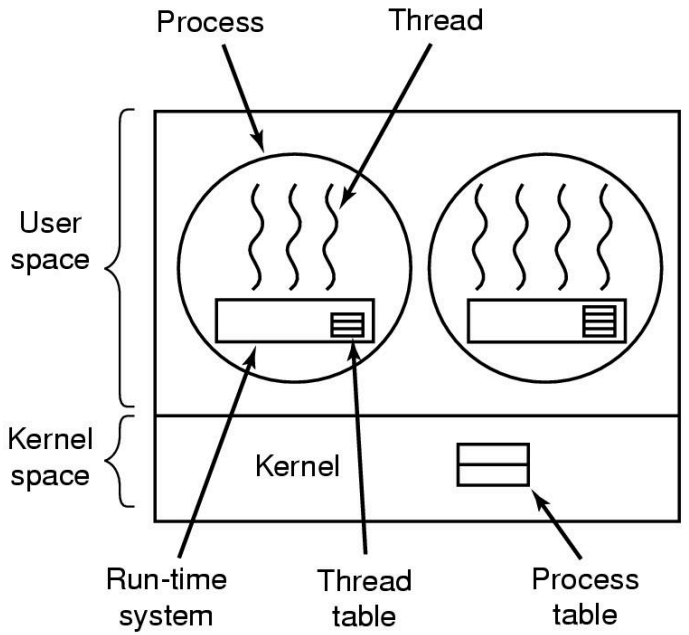
Single Process P with single thread

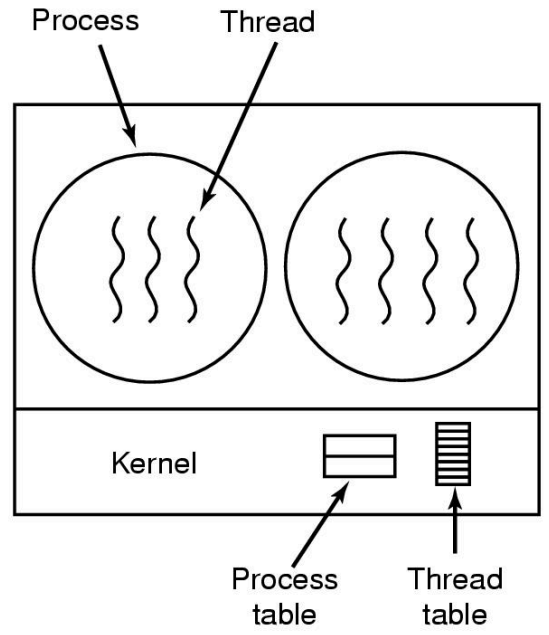


Single Process P with three threads

ZARZĄDZANIE - SCHEMATY

1. Pakiet wątków organizowany na poziomie użytkownika.
2. Pakiet wątków zarządzany przez jądro.





01

MS DOS – pojedynczy wątek sterowania,

02

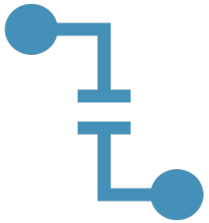
Unix – wiele procesów użytkownika, ale tylko po jednym wątku sterowania w ramach jednego procesu,

03

Solaris, Linux, Mac OSX, Windows, Tru64 obsługują wiele wątków sterowania w ramach pojedynczego procesu

SYSTEMY OPERACYJNE A WIELOWĄTKOWOŚĆ

SZEREGOWANIE – PLANOWANIE WYKONANIA PROCESÓW

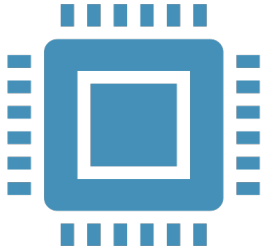


SO jednoprogramowe:

Nie wymagają planowania,

Procesy wykonują się sekwencyjnie jeden po drugim,

Okresy bezczynności procesora są stratą mocy obliczeniowej.

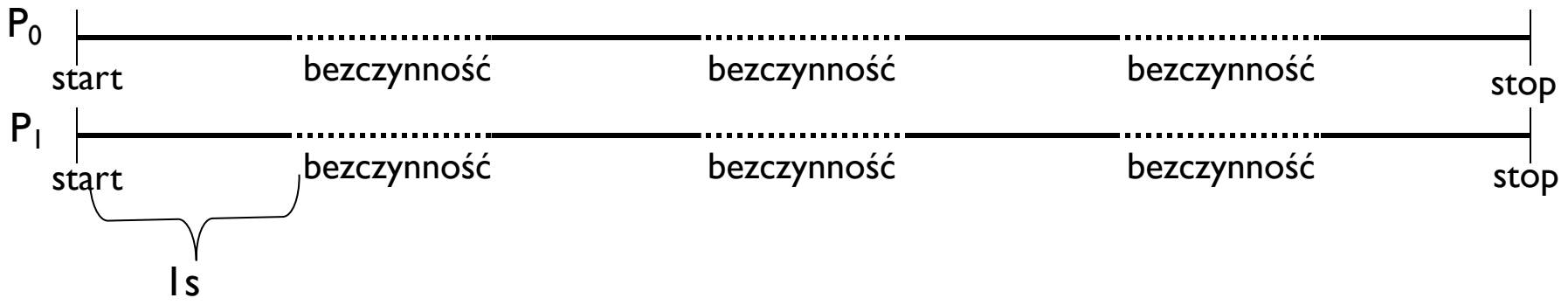


SO wieloprogramowe:

Czas procesora (nawet na maszynie jednoprocesorowej) jest dzielony między różne zadania.

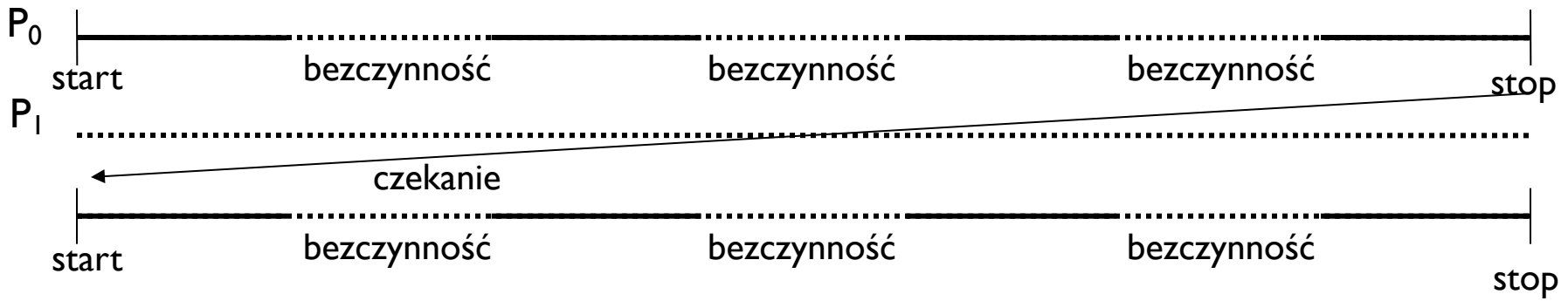
Kiedy jakiś proces musi czekać, system operacyjny odbiera mu procesor i przydziela do innego zadania.

ZAŁOŻENIA DO PRZYKŁADU



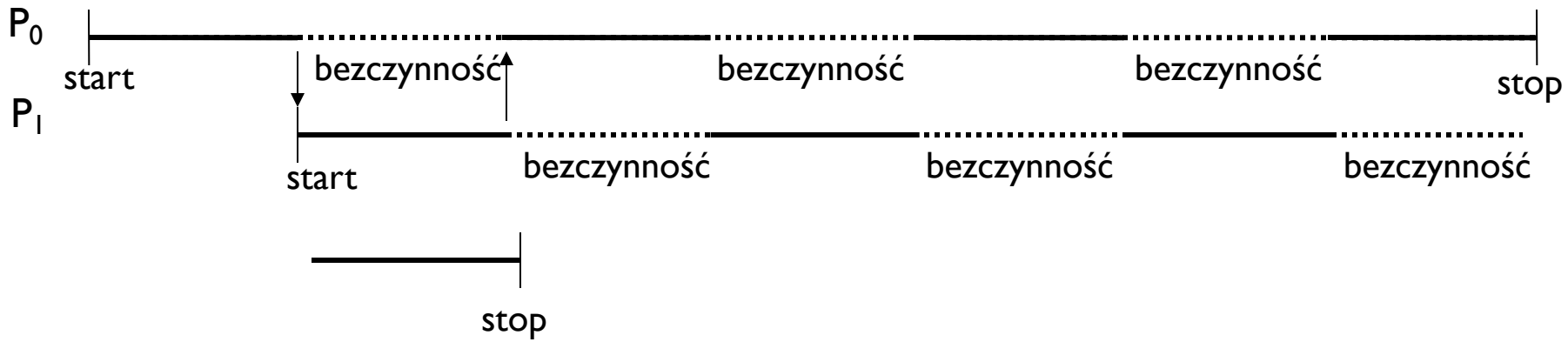
- każdy odcinek w procesie wykonywany jest przez ls
- procesy P_0 i P_1 mają być wykonane współbieżnie

PRZYKŁAD (ROZWIĄZANIE JEDNOZADANIOWE)



- P_1 czeka na wykonanie P_0
- zajętość procesora 50%
- przy 60 powtórzeniach czas wykonania 4 minuty

PRZYKŁAD (ROZWIĄZANIE WIELOZADANIOWE)



- P_1 nie czeka na wykonanie P_0
- zajętość procesora 100%
- przy 60 powtórzeniach czas wykonania 2 minuty

KOLEJKA PLANOWANIA --- SZEREGOWANIE

SO umieszcza w kolejce zadań pojawiające się procesy. Procesy oczekują tam na przydzielenie pamięci operacyjnej. Kolejka przechowywana jest w pamięci masowej.

Kolejka procesów gotowych zawiera procesy takie, które czekają na wykonanie. Kolejkę tą przechowuje się w pamięci operacyjnej.

Kolejki do urządzeń – lista procesów oczekujących na przydział urządzenia, które w danej chwili obsługuje inny proces. Każde urządzenie ma własną kolejkę.

DIAGRAM KOLEJEK W SYSTEMIE

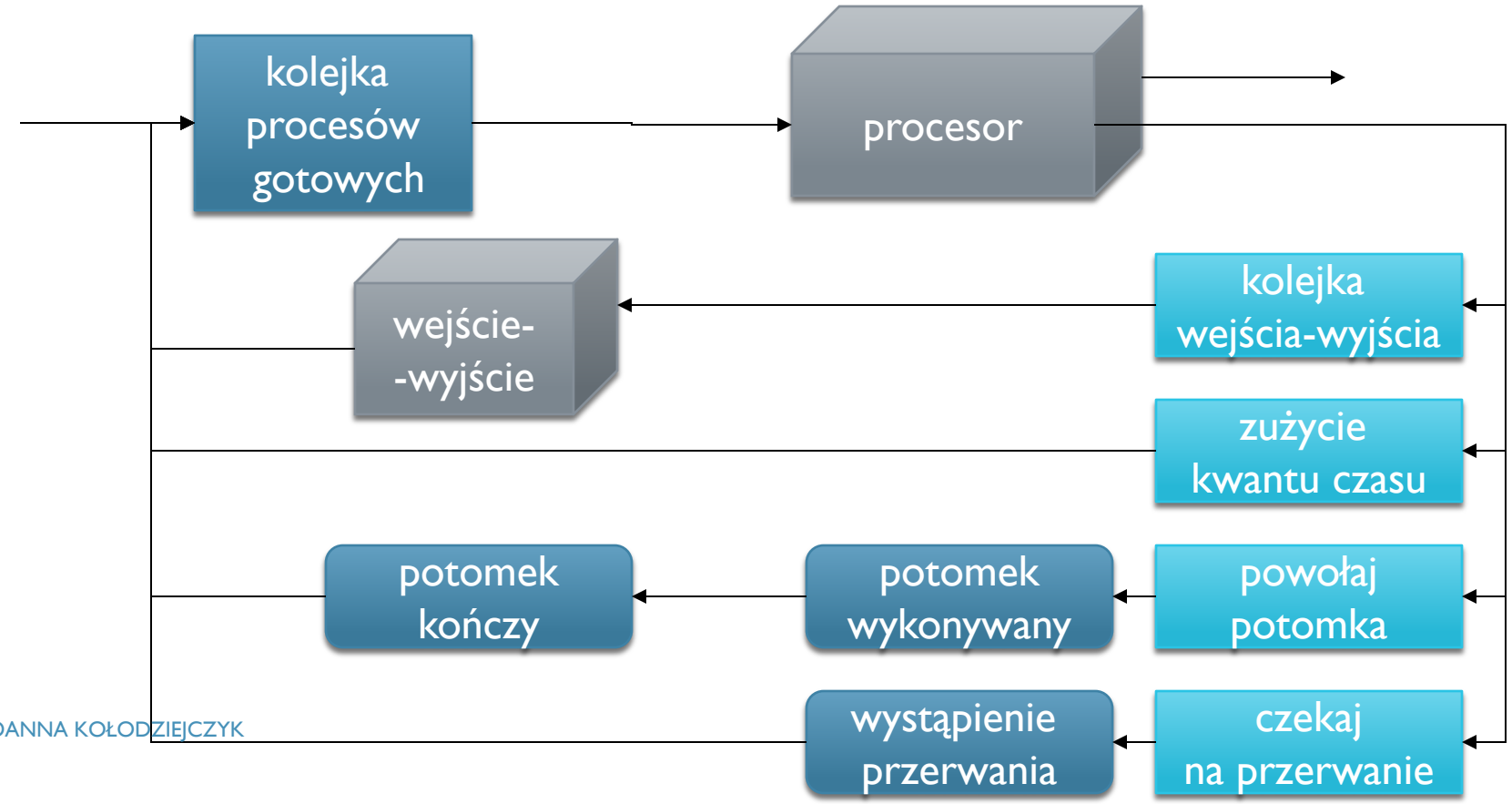


DIAGRAM KOLEJEK - OPIS

- Nowy proces na początku umieszczany jest w kolejce procesów gotowych. Oczekuje w niej do czasu wybrania go przez procesor do wykonania. Po wykonaniu może nastąpić jeden z wariantów:
  1. proces zamówi operacje wejścia-wyjścia i zostanie umieszczony w kolejce urządzenia,
  2. może utworzyć nowy proces i oczekiwać na jego zakończenie,
  3. proces przymusowo jest usunięty z procesora w wyniku przerwania i przeniesiony z powrotem do kolejki procesów gotowych.

PLANIŚCI W SO

Planista (scheduler) to proces systemu operacyjnego dokonujący wyboru procesów oczekujących w kolejce.

Planista długoterminowy (zadań) wybiera procesy z pamięci masowej. Działa rzadziej, nadzoruje stopień wieloprogramowości. Optymalny w wyborze, a nie w czasie.

Planista krótkoterminowy (przydziału procesora) wybiera procesy z kolejki procesów gotowych. Jest często wywoływany (raz na 2ms), wymaga szybkich decyzji.

PLANOWANIE TO GŁÓWNA FUNKCJA SO

Typy procesów:

- ograniczone przez **wejście-wyjście:** długie i dużo operacji na we-wy;
- ograniczone przez dostęp do **procesora**: dużo obliczeń, obciążają procesor;

Struktura planowania

- **bez wywłaszczeń**: proces po otrzymaniu czasu procesora wykonuje się, aż do zakończenia lub przejścia w stan oczekiwania;
- **wywłaszczeniowa**: możliwość przerwania wykonania procesu poprzez przerwanie systemowe.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

PRZEŁĄCZANIE KONTEKSTU

Przełączenie procesora z jednego procesu na drugi wymaga przechowania stanu starego procesu i załadowania przechowywanego stanu nowego procesu.

Program koordynujący przekazuje sterowanie procesora do procesu wybranego przez planistę.

Do jego zadań należy:

przełączanie kontekstu,	przełączanie w tryb użytkownika,	wykonanie skoku do odpowiedniego adresu w programie użytkownika w celu wznowienia działania programu.

Wykorzystanie procesora: jak najintensywniejsza eksploatacja procesora. W systemach rzeczywistych od 40% do 90%.

Przepustowość: liczba procesów kończonych w jednostce czasu.

Czas cyklu przetwarzania: czas od momentu nadejścia procesu do przetworzenia, a chwilą zakończenia. Suma okresów oczekiwania na wejście do pamięci, w kolejce procesów gotowych, wykonania i operacji wejścia- wyjścia.

Czas oczekiwania: czas spędzony w kolejce procesów gotowych.

Czas odpowiedzi: czas od złożenia zamówienia do pojawienia się odpowiedzi.

ALGORYTMY PLANOWANIA – KRYTERIA PORÓWNANIA

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

ALGORYTM FCFS

FCFS (First Come First Served) działa jak kolejka FIFO, pierwszy proces, który nadejdzie zostanie obsłużony w pierwszej kolejności. Blok kontrolny procesu dołączany jest do kolejki procesów gotowych na końcu kolejki.

Proces	Czas trwania fazy
P_1	24
P_2	3
P_3	3

diagram Gantta

OCENA FCFS

- Proces P_1 oczekuje na wykonanie 0ms, dla procesu P_2 24ms, a dla procesu P_3 27ms. Zatem średni czas oczekiwania wynosi (0+24+27)/3=17ms.
- Gdyby procesy nadeszły w kolejności P_2, P_3 , P_1 to średni czas oczekiwania wyniesie (0+3+6)/3 = 3ms.

diagram Gantta

- **Efekt konwoju** następuje gdy, duży proces zajmie procesor i wszystkie inne procesy czekają na jego zakończenie.
- FCFS jest **niewywłaszczający**. Proces kończy się poprzez wykonanie lub żądanie operacji wejścia-wyjścia.

SJF- NAJPIERW NAJKRÓTSZE ZADANIE

SJF (Shortest Job First) – przydzielanie procesu o najkrótszej fazie procesora. Jeżeli dwa procesy mają taką sama fazę stosuje się algorytm FCFS.

Proces	Czas trwania fazy
P_1	6
P_2	8
P_3	7
P_4	3

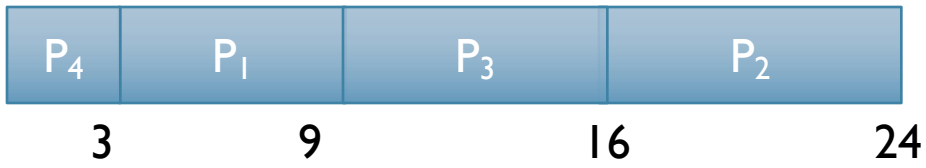


diagram Gantta

OCENA SJF

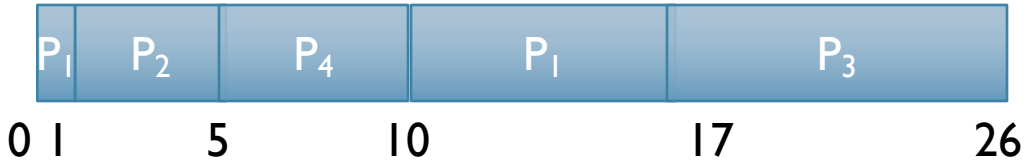
AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

- Średni czas oczekiwania dla przedstawionego przykładu (3+16+9+0)/4 = 7ms.
- Algorytm SJF jest **optymalny**. Wynika to z tego, że umieszczanie zawsze krótszych procesów przed dłuższym bardziej zmniejsza czas oczekiwania procesu krótszego niż wydłuża dłuższego.
- Problemem jest określenie długości procesu, dlatego nie nadaje się do krótkoterminowego planowania procesora.
- Algorytm może być **wywłaszczający** (gdy do kolejki wejdzie proces krótszy niż reszta obecnie wykonywanego procesu; wówczas proces bieżący jest przerywany) i **niewywłaszczający** (pozwoli zakończyć się procesowi bieżącemu).

PRZYKŁAD SJF WYWŁASZCZAJĄCEGO

Proces	Czas przybycia	Czas trwania fazy
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

diagram Gantta



średni czas oczekiwania: ((10-1)+(1-1)+(17-2)+(5-3))/4=6.5ms

PLANOWANIE PRIORYTETOWE

Każdemu procesowi przydziela się **priorytet (liczbę)**. Procesor obsługuje proces o **najwyższym** priorytecie. Procesy o tych samych priorytetach planuje się według FCFS.

Małe liczby mogą oznaczać priorytet wysoki, a duże niski lub odwrotnie.

Proces	Priorytet	Czas trwania fazy
P_1	3	10
P_2	1	1
P_3	3	2
P_4	4	1
P_5	2	5

PLANOWANIE PRIORYTETOWE - OCENA

Średni czas dla przykładu wynosi 8,2ms.

Priorytety wyznacza się na podstawie: limitów czasowych, obszaru wymaganej pamięci, liczby otwartych plików lub zewnętrznej ważności procesów, opłat wnoszonych za jego uruchomienie itp... (wewnętrzna)

Może mieć wariant **wywłaszczający** bądź **niewywłaszczający**.

Nieskończone blokowanie polega na tym, że proces o niskim priorytecie nigdy się nie wykona. Aby temu zapobiec wraz z przyrostem czasu oczekiwania (**starzenia się procesu**) zwiększa się priorytet procesu.

PLANOWANIE ROTACYJNE

Round-robin – ustala się pewną małą jednostkę czasu nazwaną **kwantem czasu** (1 do 100ms). Kolejka procesów jest cykliczna. Każdy proces oczekujący dostaje kwant czasu. Gdy się nie wykona, zrzuca kontekst i przechodzi do oczekiwania na koniec kolejki. Gdy wykona się wcześniej, to zwalnia miejsce dla kolejnego procesu.

Proces	Czas trwania fazy
P_1	24
P_2	3
P_3	3

kwant czasu = 4 ms

PLANOWANIE ROTACYJNE - OCENA

Średni czas oczekiwania wynosi 17/3 = 5.66ms.

Algorytm jest **wywłaszczający**.

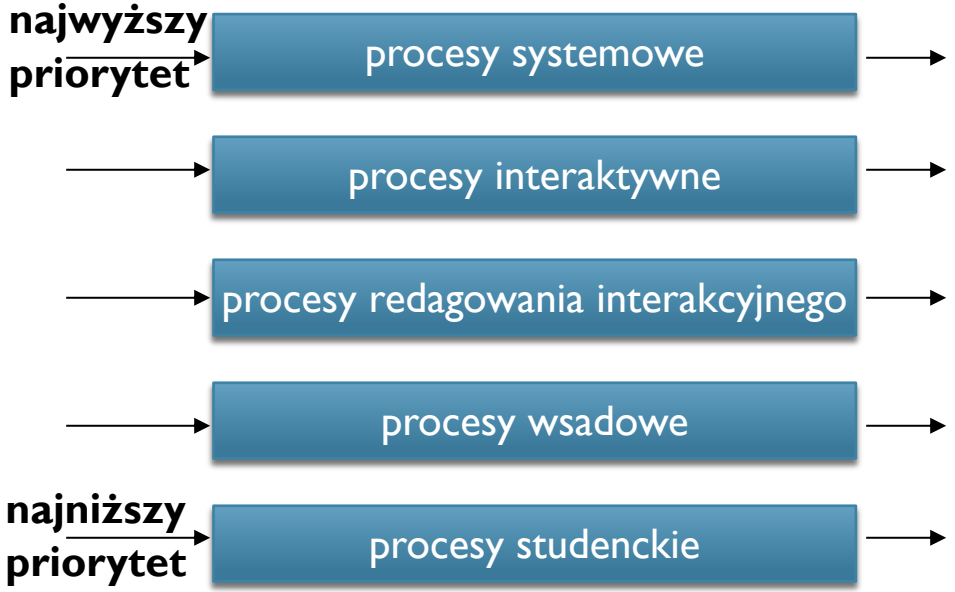
Jakość algorytmu zależy od wielkości kwantu czasu:

zbyt długi prowadzi do obsługi kolejki typu FCFS,

zbyt krótki daje efekt dzielenia procesora, jakby każdy proces miał swój procesor (tyle, że dużo wolniejszy, ze względu na przełączanie kontekstu).

WIELOPOZIOMOWE PLANOWANIE KOLEJEK

- Algorytm rozdziela kolejkę procesów gotowych na osobne kolejki w zależności od pewnych cech.
- Każda kolejka może mieć inny algorytm planowania.
- Musi istnieć planowanie pomiędzy kolejkami. Najczęściej (jak na przykładzie) jest to stało-priorytetowe planowanie wywłaszczeniowe.
- Można tez operować przydziałami czasu pomiędzy kolejkami. Np. kolejka pierwszoplanowa otrzymuje 80% czasu procesora.



PRZERWA

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

KOORDYNOWANIE PROCESÓW

Wiemy, że: współbieżność przetwarzania jest podstawową cechą wieloprogramowych systemów operacyjnych.

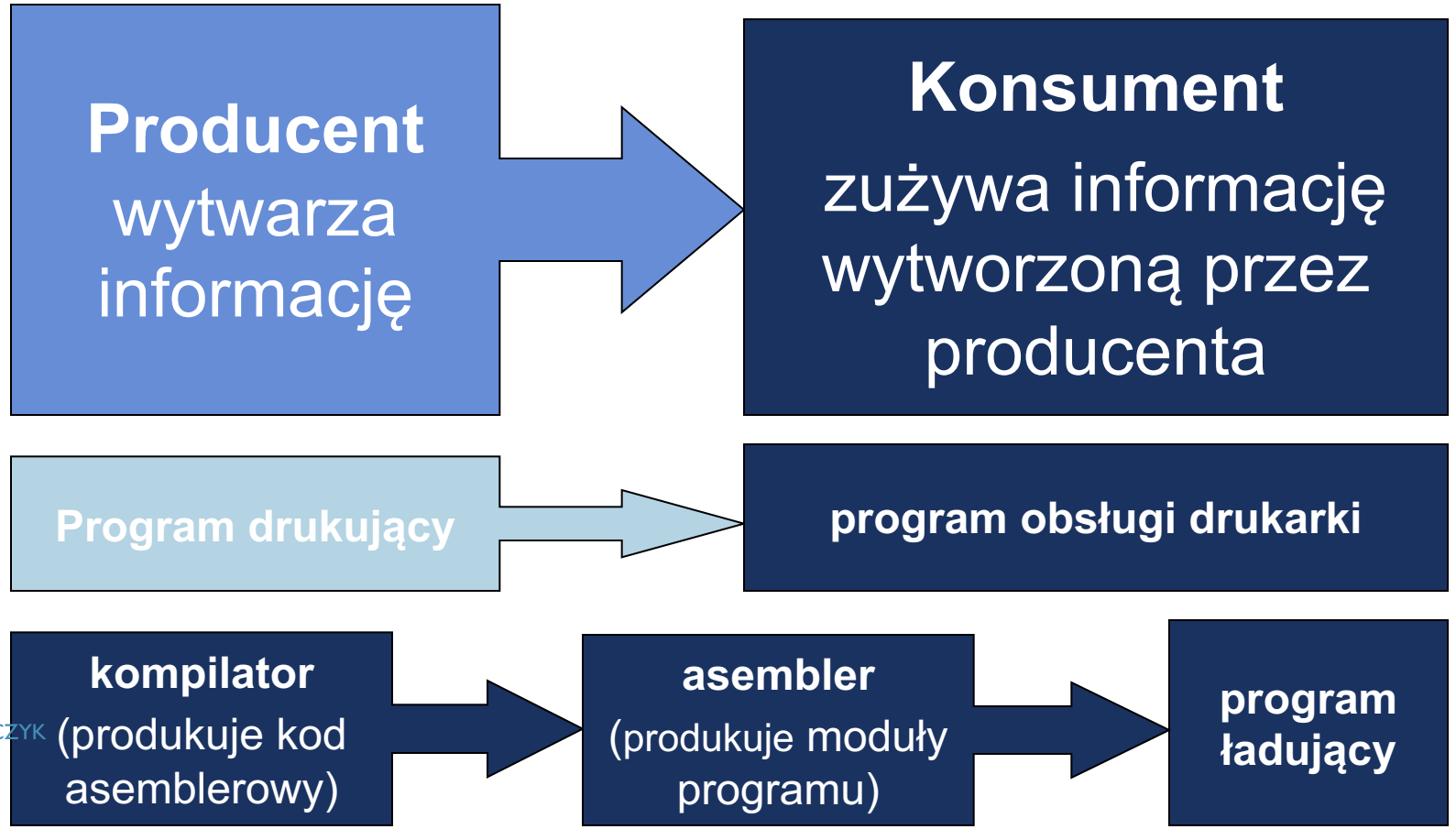
Zakładamy, że system współbieżny składa się z procesów:

- systemu operacyjnego,
- użytkowych (użytkownika),

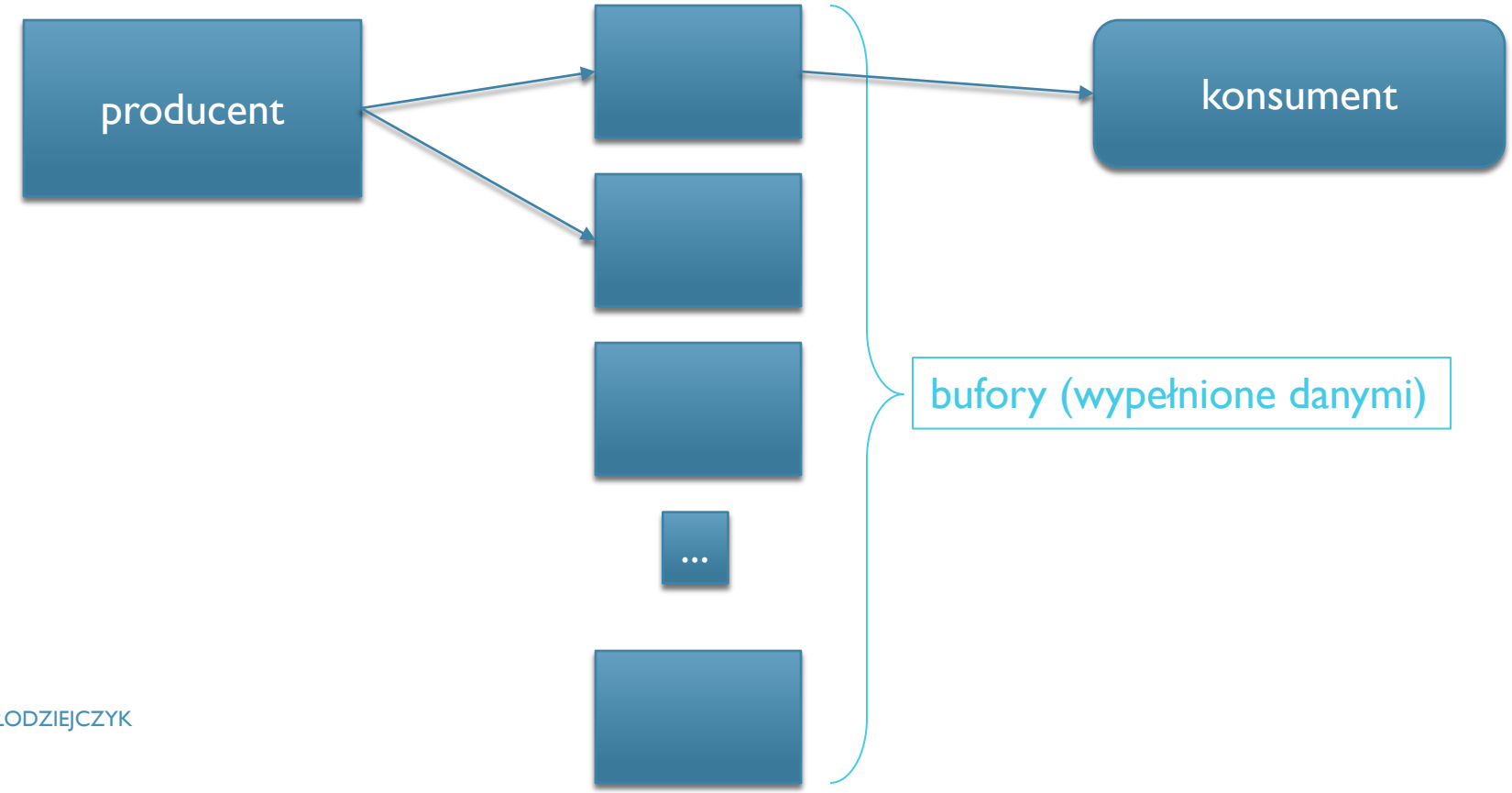
Aby procesy wykonały się w porządku należy stosować mechanizmy synchronizacji i komunikacji pomiędzy procesami.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

PROCESY PODZIAŁ NA PRODUCENT-KONSUMENT



BUFORY DLA PRODUCENTA I KONSUMENTA



ZAŁOŻENIA POPRAWNEGO DZIAŁANIA UKŁADU

producent wypełnia bufory produkowanymi danymi,

konsument opróżnia bufory z danych,

producent może wysyłać wyniki do innego bufora niż ten, z którego pobiera konsument,

procesy producenta i konsumenta muszą być zsynchronizowane (konsument nie może przeczytać czegoś, czego producent nie wytworzył),

konsument oczekuje na odpowiednie dla niego dane.

BENCHMARKI

Metody synchronizacji testowane są na następujących problemach:

- ograniczonego bufora,
- czytelników i pisarzy,
- posilających się filozofów.

DWA WARIANTY BUFOROWANIA

nieograniczony bufor

- nie ma praktycznych ograniczeń na liczbę buforów,
- konsument może czekać na jednostki od producenta,
- producent produkuje nieustannie, zawsze istnieją puste bufory,

ograniczony bufor

- liczba buforów ustalona,
- konsument czeka, jeżeli wszystkie bufory są puste,
- producent czeka, jeżeli wszystkie bufory zapełnione.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

ROZWIĄZANIE PROBLEMU OGRANICZONEGO BUFORA (PRODUCENT)

1. Produkuje jednostkę danych do bufora;
2. Jeżeli wszystkie bufory zajęte, czekaj i sprawdzaj czy któryś się nie zwolni;
3. Gdy jest pusty bufor, to wypełnij go danymi;
4. Pobierz numer kolejnego bufora do wypełnienia (jeżeli bieżący bufor jest ostatni, to kolejny do wypełnienia będzie pierwszy (lista cykliczna));
5. Zwiększ liczbę zajętych buforów o 1.

ROZWIĄZANIE PROBLEMU OGRANICZONEGO BUFORA (KONSUMENT)

Jeżeli wszystkie bufory są puste, to czekaj i sprawdzaj czy nie pojawi się jakiś wypełniony bufor;

Gdy bufor jest pełen, pobierz jednostkę danych z odpowiedniego bufora;

Pobierz numer kolejnego bufora do pobrania (jeżeli bieżący bufor jest ostatni, to kolejny do pobrania będzie pierwszy (lista cykliczna));

Liczbę zajętych buforów zmniejsz o 1;

Skonsumuj (wykorzystaj) jednostkę pobranych danych.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

WADY ROZWIĄZANIA

Niepoprawne działania przy realizacji współbieżnej, przykład:

1. zakładamy, że liczba zajętych buforów = 5,
2. producent wykonuje zwiększenie liczby buforów zajętych o 1,
3. konsument wykonuje zmniejszenie liczby buforów zajętych o 1 (w tym samym czasie co producent),
4. po wykonaniu punktów 2 i 3 liczba zajętych buforów może wynosić: 4, 5 lub 6, gdy poprawny jest tylko 5.

```
t0: producent wykonuje rejestr1:=licznik    [rejestr1 = 5]
t1: producent wykonuje rejestr1:=rejestr1+1    [rejestr1 = 6]
t2: konsument wykonuje rejestr2:=licznik    [rejestr2 = 5]
t3: konsument wykonuje rejestr2:=rejestr2-1    [rejestr2 = 4]
t4: producent wykonuje licznik:=rejestr1    [licznik = 6]
t5: konsument wykonuje licznik:= rejestr2   [licznik = 4]
```

ANALIZA BŁĘDU WSPÓŁBIEŻNOŚCI

Wykonanie przykładowej procedury powoduje przekłamanie w stanie licznika, choć zajętych jest 5 buforów wskazanie jest na 4;

Problem leży w tym, że zarówno producent jak i konsument manipulują wartością liczby buforów zapełnionych;

Rozwiązanie: tylko jeden proces w danej chwili może wykonywać zmiany w wartości liczby buforów – takie zabezpieczenie nazywane jest synchronizacją i koordynacją procesów.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

ROZWIĄZANIE 1- SEKCJA KRYTYCZNA

- Założenia
  - system składa się z *n* procesów $\{P_1, P_2,...,P_n\}$,
  - każdy proces ma segment kodu nazwany **sekcją krytyczną** (wspólne zmienne, pisanie plików, aktualizacja tablic).
- Cechy sekcji krytycznej
  - tylko jeden proces wykonuje sekcję – wzajemne wyłączenie w czasie;
  - organizacja współpracy procesów w obrębie sekcji krytycznej
    - każdy proces prosi o pozwolenie wejścia do sekcji krytycznej – **sekcja wejściowa**
    - po wykonaniu sekcji krytycznej wykonywana jest **sekcja wyjściowa**
    - pozostały kod to tzw. **reszta.**

WARUNKI ROZWIĄZANIA PROBLEMU SEKCJI KRYTYCZNEJ

Wzajemne wyłączanie:

- Jeżeli proces P działa w **sekcji krytycznej** to żaden inny proces nie działa w sekcji krytycznej.

Postęp:

- Jeżeli żaden proces nie działa w sekcji krytycznej, a istnieją procesy **oczekujące** na wejście do niej, to tylko procesy nie wykonujące swoich **reszt** mogą kandydować do wejścia do sekcji i jednocześnie wybór nie może być odwlekany w nieskończoność.

Ograniczone czekanie

- Musi istnieć skończona liczba wejść innych procesów do sekcji krytycznej, po tym, jak dany proces zgłosił chęć wejścia do sekcji.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

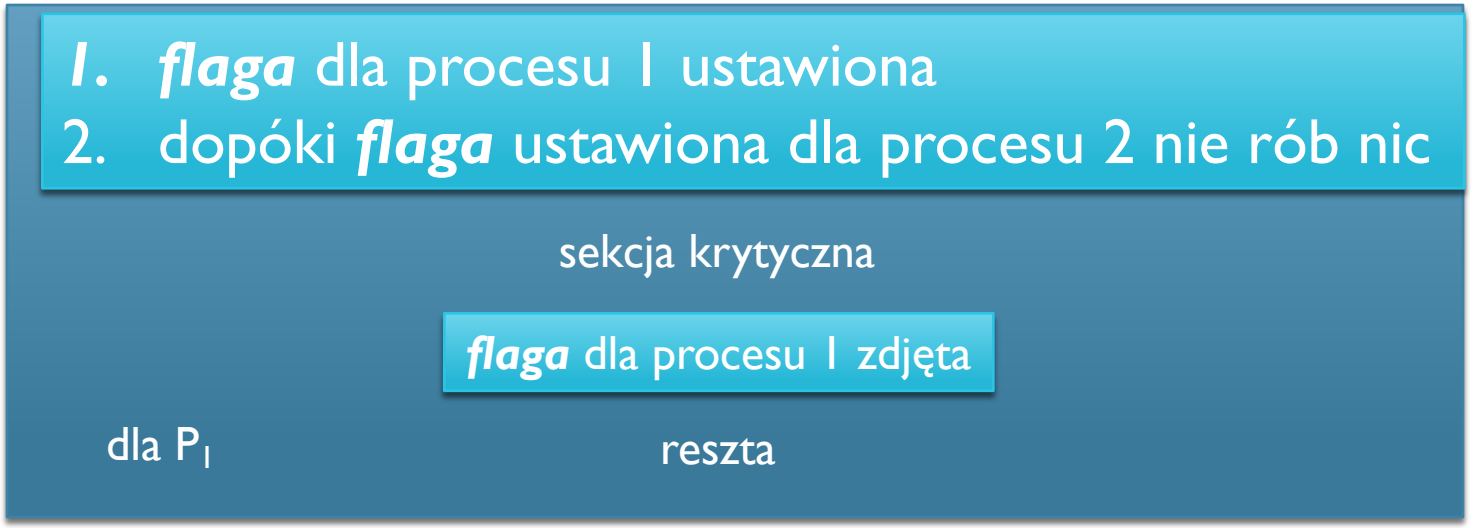
ISTOTNE ZAŁOŻENIA PRZED ANALIZĄ ALGORYTMÓW SYNCHRNIZACJI

- procesy mają niezerową prędkość wykonania,
- podstawowe rozkazy maszynowe (umieść, pamiętaj, sprawdź) wykonywane są niepodzielnie (jeżeli dwa takie rozkazy są wykonywane współbieżnie, to faktycznie wykonane są sekwencyjnie w nieznanej kolejności),
- oznaczenie struktury procesu:

ROZWIĄZANIA DLA 2 PROCESÓW P_1 I P_2 (ALGORYTM 1)

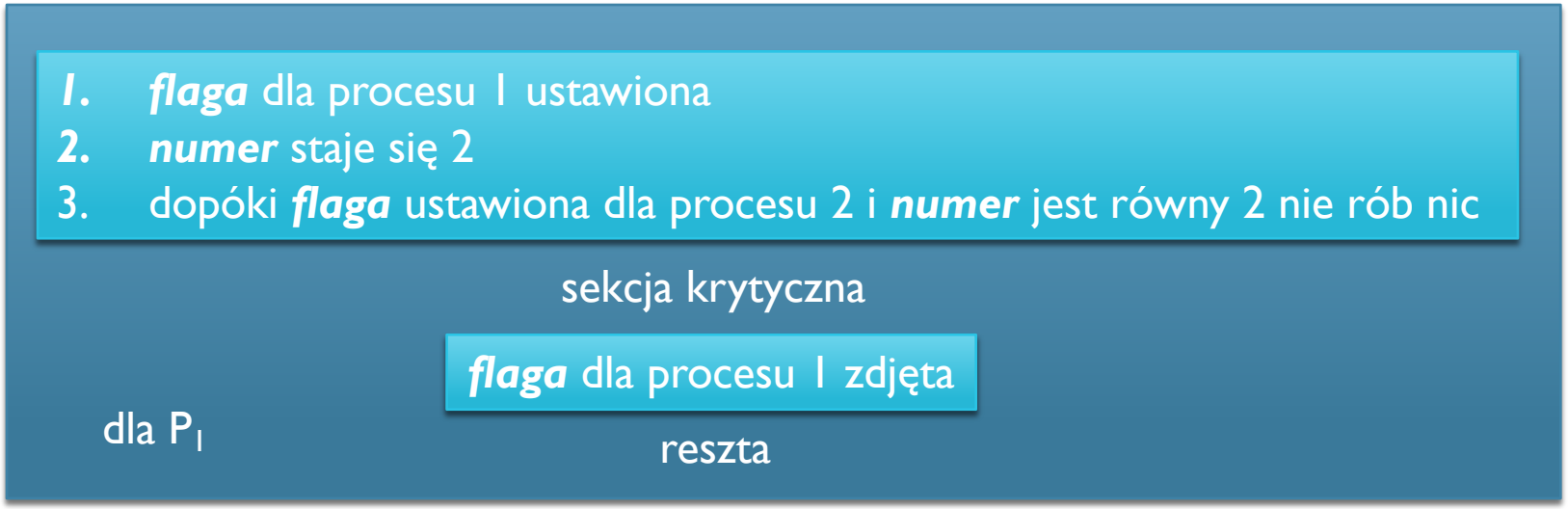
- ***numer*** (procesu)– zmienna dzielona przez dwa procesy
- Analiza:
  - zapewnia, że tylko jeden proces znajduje się w sekcji krytycznej;
  - nie zapewnia postępu (procesy mogą wykonywać sekcje tylko naprzemiennie).

ROZWIĄZANIA DLA 2 PROCESÓW P_1 I P_2 (ALGORYTM 2)



- ***flaga*** – jest ustawiana gdy proces gotowy do wejścia do sekcji krytycznej
- Analiza:
  - zapewnia, warunek wzajemnego wyłączania;
  - nie zapewnia postępu np. oba procesy podniosą flagę gotowości wejścia do sekcji krytycznej i nigdy do niej nie wejdą, bo utkną w sprawdzaniu warunku gotowości flagi.

ROZWIĄZANIA DLA 2 PROCESÓW P_1 I P_2 (ALGORYTM 3)



- ***numer*** – zmienna dzielona przez dwa procesy
- ***flaga*** – jest ustawiana gdy proces gotowy do wejścia do sekcji krytycznej

ANALIZA ALGORYTMU TRZECIEGO

Ustawienie wartości ***numer*** na 2 oznacza, że przypuszczamy, że drugi proces też chce wejść do sekcji krytycznej, o ile będzie to możliwe.

Jeżeli oba procesy próbują wejść do sekcji w tym samym czasie, i będą zmieniać wartość ***numer***, to i tak podstawienie na poziomie kodu maszynowego jest sekwencyjne. Ostatecznie tylko jeden z procesów będzie mógł wejść do sekcji krytycznej w pierwszej kolejności.

Warunek wzajemnego wyłączania

- proces wchodzi do sekcji gdy ***flaga*** drugiego procesu jest opuszczona **albo *numer*** jest równy numerowi bieżącego procesu.
- nawet gdy obie ***flagi*** są podniesione, to ***numer*** może mieć tylko jedną z wartości, więc niemożliwe jest wejście obu procesów do sekcji krytycznej.

Warunek postępu i warunek ograniczonego czekania

- proces P_1 utknie w pętli sprawdzania warunków i nie zrobi postępu gdy ***flaga*** dla P_2 ustawiona, a ***numer*** ustalony na 2. Taki stan nie jest możliwy do utrzymania, bo:
  - jeżeli P_2 nie chce wejść do sekcji krytycznej to ***flaga*** opuszczona (P_1 wchodzi do sekcji);
  - jeżeli P_2 jest w sekcji to wychodząc opuszcza swoja ***flagę*** (P_1 wchodzi do sekcji);
  - jeżeli P_2 podniesie swoja ***flagę*** to ***numer*** może mieć tylko 1 wartość (wykona się sekcja dla P_2 lub P_1, a w następnej kolejności sekcja dla przeciwnego procesu) zatem oczekiwanie na wykonanie sekcji krytycznej dla procesu bieżącego jest nie dłuższe niż wykonanie sekcji dla procesu drugiego.

WYKAZANIE POPRAWNOŚCI ALGORYTMU TRZECIEGO

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

SEMAFORY – METODA SYNCHRONIZACJI

- Algorytm z flagą i numerem jest trudny do uogólnienia w złożonych zagadnieniach, dlatego innym rozwiązaniem sekcji krytycznej jest **SEMAFOR**.
- Semafor **S**, który jest liczbą całkowitą jest dostępny tylko za pomocą dwóch operacji:
  - czekaj(S):
    1. dopóki S jest mniejsze lub równe 0 nie rób nic
    2. zmniejsz wartość S o 1
  - sygnalizuj(S)
    1. zwiększ wartość S o 1
- Zmiany wartości semafora [czekaj i sygnalizuj] są niepodzielne, atomowe, a nadto podczas instrukcji czekaj nie może nastąpić przerwanie systemowe.

SPOSOBY STOSOWANIA SEMAFORÓW

- Stosuje się je do rozwiązania problemu sekcji krytycznej dla **n** procesów.
- Semafor *wzajemnego wyłączania* ***wzajwy*** (ang. mutex) jest dzielony przez wszystkie procesy i na początku ma wartość **1**.
- Proces ma następująca postać:

PRZYKŁAD SYNCHRONIZACJI Z SEMAFORAMI

- Proces P2 ma się wykonać tylko wtedy, gdy zakończy się wykonanie procesu P1.
- **synch** to współdzielona wartość równa na początku 0.

proces P1:	proces P2:
operacje	**czekaj(synch)**
sygnalizuj(synch)	operacje

- Proces P2 będzie tak długo czekał na wykonanie, aż **synch** będzie miało wartość większą od 0.
- **synch** zyska wartość większą od 0 tylko gdy wykonają się operacje z P1.

IMPLEMENTACJA SEMAFORA

- Semafor pamięta swoją **wartość** i całą **listę procesów czekających** pod danym semaforem.
- Operacja **sygnalizuj**, zwiększa wartość semafora, usuwa jeden proces z listy oczekujących procesów i budzi go.
- Operacja **oczekiwania** zmniejsza wartość semafora i jednocześnie dołącza proces do listy oraz zablokowuje proces.
- **Wartość ujemna semafora** oznacza, że w kolejce pod semaforem znajdują się procesy do wzbudzenia (wartość bezwzględna wskazuje liczbę procesów w liście).
- Do obsługi kolejki semafora można zastosować np. FIFO (pierwszy wchodzący do listy pierwszy z niej wychodzi).
- !!!!nie można wykonać zmiany semafora w tym samym czasie przez dwa procesy. Operacje na semaforach musza być **atomowe**!!!!
- Implementacja semaforów z kolejką może prowadzić do **blokad** – np.. dwa procesy czekają na sygnalizację od siebie nawzajem (omówione później).
- https://www.youtube.com/watch?v=PQ5aK5wLCQE

PROBLEM OGRANICZONEGO BUFORA NA SEMAFORACH

- Buforów jest n, a każdy mieści 1 jednostkę.
- Producent wprowadza (wypełnia) dane do buforów, gdy konsument z buforów dane pobiera (opróżnia).

Rozwiązanie za pomocą semaforów.

pusty = n

pełny = 0

wzajwy = 1

producent

produkuje jednostkę
...
czekaj(pusty)
czekaj(wzajwy)
...
dodaj jednostkę do bufora
...
sygnalizuj(wzajwy)
sygnalizuj(pełny)

konsument

czekaj(pełny)
czekaj(wzajwy)
...
wyjmij jednostkę z bufora
...
sygnalizuj(wzajwy)
sygnalizuj(pusty)
...
konsumowanie jednostki

KOMUNIKACJA MIĘDZYPROCESOWA

- Omówione zagadnienia problemu synchronizacji składają się na szerszy problem komunikacji pomiędzy procesami, które mają ze sobą współpracować.
- Istnieją dwa schematy komunikacji:
  1. pamięć dzielona – współużytkowanie pewnych zmiennych (np. ograniczony bufor); system operacyjny musi dostarczać środki do dzielenia pamięci,
  2. system komunikatów – procesy wymieniają komunikaty, wymianę zapewnia system operacyjny.

Oba schematy mogą być stosowane jednocześnie.

SYSTEM KOMUNIKATÓW

Nie stosuje zmiennych dzielonych.

Wykorzystuje dwie podstawowe operacje:

- nadaj(komunikat),
- odbierz(komunikat).

Komunikaty mogą mieć

- stałą długość: łatwa fizyczna implementacja, ograniczone programowanie,
- zmienną długość: trudna fizyczna implementacja, łatwe programowanie.

Łącze komunikacyjne – do zadawania i odbierania komunikatów pomiędzy procesami P i Q.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

CECHY ŁĄCZA KOMUNIKACYJNEGO NA POZIOMIE LOGICZNYM

łącze ustalane jest pomiędzy dwoma lub więcej procesami;

liczba łącz pomiędzy procesami jest określona;

pojemność łącza: buforowane lub nie;

długość komunikatów: zmienna lub stała;

jednokierunkowe to takie gdzie proces może tylko nadawać lub odbierać (może łączyć więcej niż dwa procesy) lub dwukierunkowe łącze, gdzie proces może odbierać i nadawać komunikaty, lub łącze ma przynajmniej jeden proces odbiorczy;

komunikacja bezpośrednia lub pośrednia;

komunikacja symetryczna lub asymetryczna;

buforowanie automatyczne lub jawne.

AUTOR: DR INŻ. JOANNA KOŁODZIEJCZYK

KOMUNIKACJA BEZPOŚREDNIA

- Proces, który chce nadać lub odebrać komunikat musi jawnie nazwać **odbiorcę** lub **nadawcę.**
- Komunikaty mają postać (wersja **symetryczna**):
  - nadaj(P,komunikat): nadaj komunikat do P
  - odbierz(P,komunikat): odbierz komunikat od P.
- Komunikaty mają postać (wersja **asymetryczna**):
  - nadaj(P,komunikat): nadaj komunikat do P
  - odbierz(id,komunikat): odbierz komunikat od dowolnego procesu; pod id zostanie wstawiona nazwa procesu, od którego nadszedł komunikat.
- **Właściwości** łącza
  - ustanawiane automatycznie pomiędzy dwoma procesami, do ustanowienia komunikacji wystarczy by znały swoje identyfikatory,
  - łącze dotyczy 2 procesów,
  - pomiędzy parą istnieje dokładnie 1 łącze,
  - łącze jest dwukierunkowe.

REALIZACJA PROBLEMU PRODUCENT-KONSUMENT ŁĄCZEM BEZPOŚREDNIM

producent

produkuje jednostkę
...
nadaj(konsument, jednostka)

konsument

odbierz(producent, jednostka)
...
konsumuje jednostkę
...

Wady łącza: zmiana nazwy jednego procesu musi pociągać weryfikację wszystkich nazw w odwołaniach do procesów.

KOMUNIKACJA POŚREDNIA

- **Porty**: rodzaj skrzynek pocztowych, do których kieruje się komunikaty i z których się komunikaty odbiera.
- Port ma identyfikator np. 8080.
- Z wybranym procesem można komunikować się za pomocą różnych skrzynek pocztowych.
- Komunikaty mają postać (wersja symetryczna):
  - nadaj(A,komunikat): nadaj komunikat do skrzynki A
  - odbierz(A,komunikat): odbierz komunikat ze skrzynki A.
- Właściwości łącza:
  - łącze ustanawiane, gdy procesy dzielą skrzynkę,
  - łącze może być ustanowione pomiędzy więcej niż dwoma procesami,
  - każda para może mieć kilka różnych łączy, z których każde odpowiada jakiejś skrzynce pocztowej,
  - łącze może być jedno- i dwukierunkowe.

CECHY PORTU

- Port może być własnością:
  - procesu (przypisana i definiowana jako cześć procesu):
    - port posiada właściciela (odbiera z niej komunikaty);
    - i użytkownika (nadaje komunikaty do portu);
    - proces właściciel kończąc działanie usuwa port; odwoływanie się do niego po usunięciu zgłasza do procesu nadawcy komunikat o sytuacji wyjątkowej;
  - systemu operacyjnego
    - port jest niezależny i nie jest przypisany do żadnego procesu;
    - SO dostarcza mechanizmy by: tworzyć i likwidować port; nadawać i odbierać komunikaty za jego;
    - port tworzony jest na zamówienie procesu, który staje się jego właścicielem. Można jednak własność przepisać do innego procesu za pomocą funkcji SO.

BUFOROWANIE W KOMUNIKACJI PROCESÓW

- Łącze komunikacyjne może mieć pewną pojemność, określającą liczbę komunikatów mogących w nim przebywać. Jest to rodzaj kolejki komunikatów
  - pojemność **zerowa**: łącze nie dopuszcza, by jakieś komunikaty w nim oczekiwały, aby przesłać komunikat trzeba zsynchronizować procesy (rendez-vous);
  - pojemność **ograniczona**: długość n – mieści n komunikatów, nadawca nie musi czekać na odebranie komunikatu tylko umieszcza jego kopie lub wskaźnik do niego w kolejce (chyba, że kolejka jest wypełniona);
  - pojemność **nieograniczona**: oczekuje tu dowolna liczba komunikatów i nadawca nigdy nie jest opóźniony.

BLOKADY

Deadlock – kiedy procesy rywalizują o te same zasoby i wzajemnie je przetrzymując nie mogąc prowadzić do kontynuacji obliczeń.

System dysponuje zasobami (cykle procesora, pamięć, urządzenia wejścia-wyjścia) współdzielonymi pomiędzy procesami. Proces musi zamówić zasób, przed jego użyciem, by na koniec go zwolnić. Nie można zamówić więcej zasobów niż jest w systemie.

KOLEJNOŚĆ UŻYCIA ZASOBU PRZEZ PROCES:

DEADLOCK

Przydzielanie i zwalnianie zasobów realizuje się za pomocą funkcji systemowych , semaforów. W tablicy systemowej pamięta się stan zasobu i proces, który go wykorzystuje.

Zbiór procesów pozostaje w blokadzie, jeżeli każdy proces z tego zbioru czeka na zdarzenie, które może być spowodowane tylko przez inny proces z tego zbioru.

CHARAKTERYSTYKA BLOKADY

Blokada powstanie tylko wtedy gdy są spełnione warunki:

- wzajemne wyłączanie: przynajmniej 1 zasób musi być niepodzielny (1 proces może go używać w danej jednostce czasu);
- przetrzymywanie i oczekiwanie: musi istnieć proces przetrzymujący zasób i oczekujący na zwolnienie innego zasobu;
- brak wywłaszczenia: zasób może być zwolniony tylko z inicjatywy przetrzymującego go procesu;
- czekanie cykliczne: zbiór procesów oczekuje na zwolnienie zasobu przez jeden z procesów z rozpatrywanego zbioru.

GRAF PRZYDZIAŁU ZASOBÓW

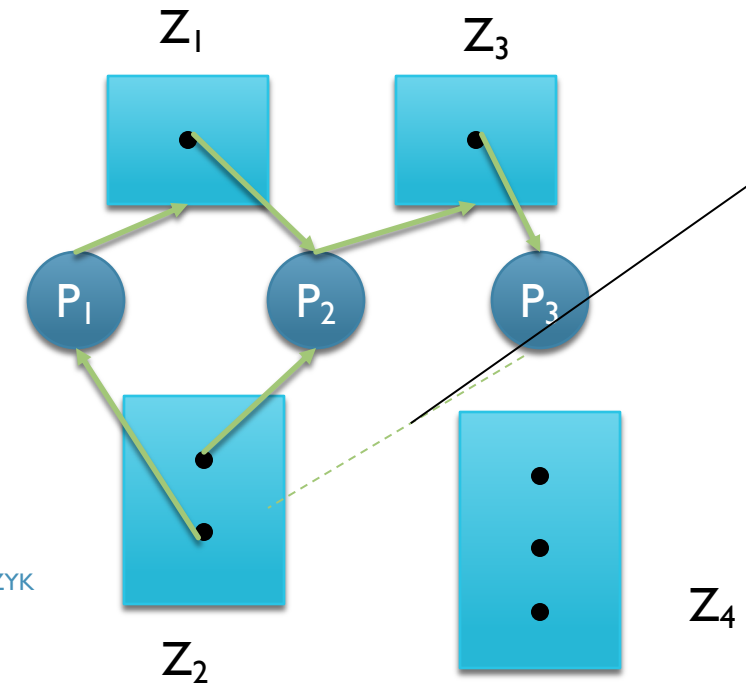
- Graf skierowany składa się z wierzchołków i krawędzi.
- Wierzchołkami mogą być procesy (P_i) lub zasoby (Z_j).
- Krawędź skierowana od procesu do zasobu oznacza, że proces zamówił egzemplarz zasobu – krawędź zamówienia.
- Krawędź skierowana od zasobu do procesu oznacza, że zasób jest przydzielony do procesu – krawędź przydziału.
- Graficznie proces prezentowany w postaci kółka, zasób z postaci prostokąta. Każdy zasób może mieć kilka egzemplarzy (np.. kilka drukarek) oznaczone jako kropki w prostokącie zasobu.

PRZYKŁAD

$P=\{P_1, P_2, P_3\}$

$Z=\{Z_1, Z_2, Z_3, Z_4\}$

$K=\{P_1 \rightarrow Z_1, P_2 \rightarrow Z_3, Z_1 \rightarrow P_2, Z_2 \rightarrow P_2, Z_2 \rightarrow P_1, Z_3 \rightarrow P_3\}$



Żądanie zasobu 2 przez proces 3 spowoduje blokadę. Blokada jest jednoznaczna z istnieniem cyklu w grafie. W rozpatrywanym przypadku są dwa cykle.
$P_1 \rightarrow Z_1 \rightarrow P_2 \rightarrow Z_3 \rightarrow P_3 \rightarrow Z_2 \rightarrow P_1$
$P_2 \rightarrow Z_3 \rightarrow P_3 \rightarrow Z_2 \rightarrow P_2$