Systemy Operacyjne (5)

Marcin Gogolewski
marcing@wmi.amu.edu.pl

Uniwersytet im. Adama Mickiewicza w Poznaniu

Poznań, 15 grudnia 2018

Rodzaje urządzeń

Blokowe

Adresowane są bloki danych, odczyt możliwy w blokach, dostępna operacja `seek()`, bloki zwykle są potęgami liczby 2 (np. 512, 1024, 2048, ...). Np. dyski.

Znakowe

Brak operacji `seek`, dostęp do kolejnych bajtów (np. mysz, klawiatura, karta sieciowa, drukarka – można wysłać większy pakiet danych, ale nie ma np. narzuconej struktury, czy rozmiaru).

Klasyfikacja nie jest doskonała, np. są urządzenia, które generują tylko przerwania (np. zegary).

Rodzaje urządzeń

Blokowe

Adresowane są bloki danych, odczyt możliwy w blokach, dostępna operacja `seek()`, bloki zwykle są potęgami liczby 2 (np. 512, 1024, 2048, ...). Np. dyski.

Znakowe

Brak operacji `seek`, dostęp do kolejnych bajtów (np. mysz, klawiatura, karta sieciowa, drukarka – można wysłać większy pakiet danych, ale nie ma np. narzuconej struktury, czy rozmiaru).

Klasyfikacja nie jest doskonała, np. są urządzenia, które generują tylko przerwania (np. zegary).

Rodzaje urządzeń

Blokowe

Adresowane są bloki danych, odczyt możliwy w blokach, dostępna operacja `seek()`, bloki zwykle są potęgami liczby 2 (np. 512, 1024, 2048, ...). Np. dyski.

Znakowe

Brak operacji `seek`, dostęp do kolejnych bajtów (np. mysz, klawiatura, karta sieciowa, drukarka – można wysłać większy pakiet danych, ale nie ma np. narzuconej struktury, czy rozmiaru).

Klasyfikacja nie jest doskonała, np. są urządzenia, które generują tylko przerwania (np. zegary).

Przepustowość wejścia/wyjścia

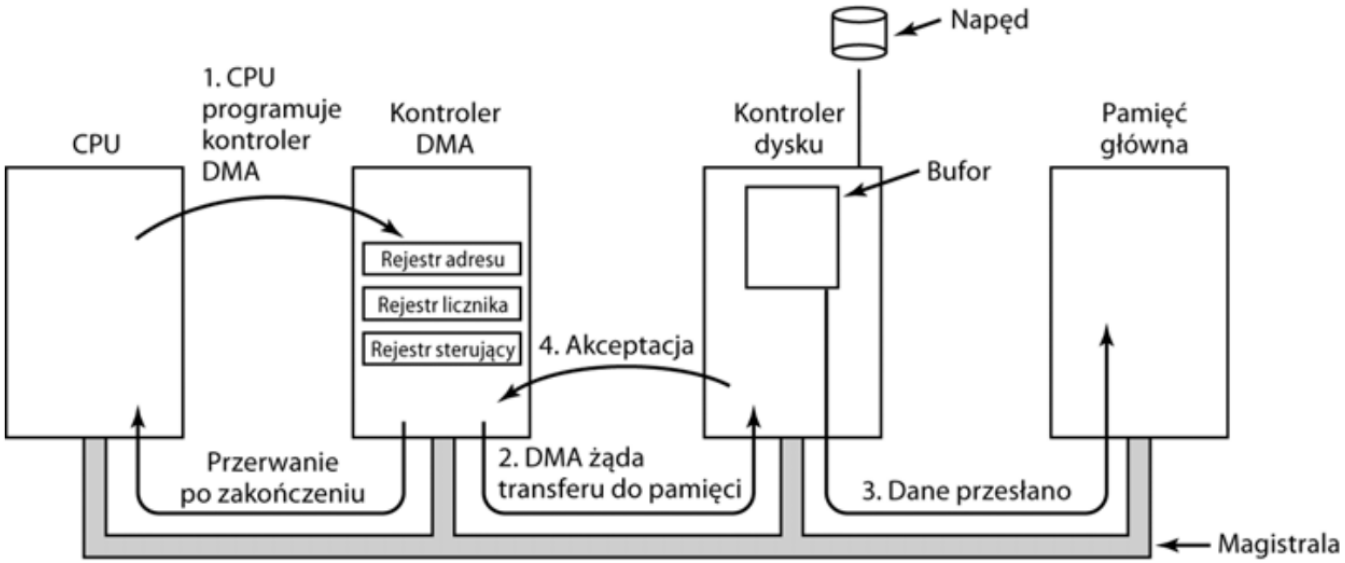
Urządzenie	Szybkość przesyłania danych
Klawiatura	10 bajtów/s
Mysz	100 bajtów/s
56K modem	7 kB/s
Skaner pracujący w rozdzielczości 300 dpi	1 MB/s
Kamera cyfrowa	3,5 MB/s
Dysk Blu-ray o szybkości 4x	18 MB/s
Sieć bezprzewodowa 802.11n	37,5 MB/s
USB 2.0	60 MB/s
FireWire 800	100 MB/s
Sieć Gigabit Ethernet	125 MB/s
Napęd dysku SATA 3	600 MB/s
USB 3.0	625 MB/s
Magistrala SCSI Ultra 5	640 MB/s
Jednopasmowa magistrala PCIe 3.0	985 MB/s
Magistrala Thunderbolt 2	2,5 GB/s
Sieć SONET OC-768	5 GB/s

Dostęp do urządzeń

- porty wejścia/wyjścia (oddzielna przestrzeń)
- mapowanie w pamięci (ta sama przestrzeń adresowa)

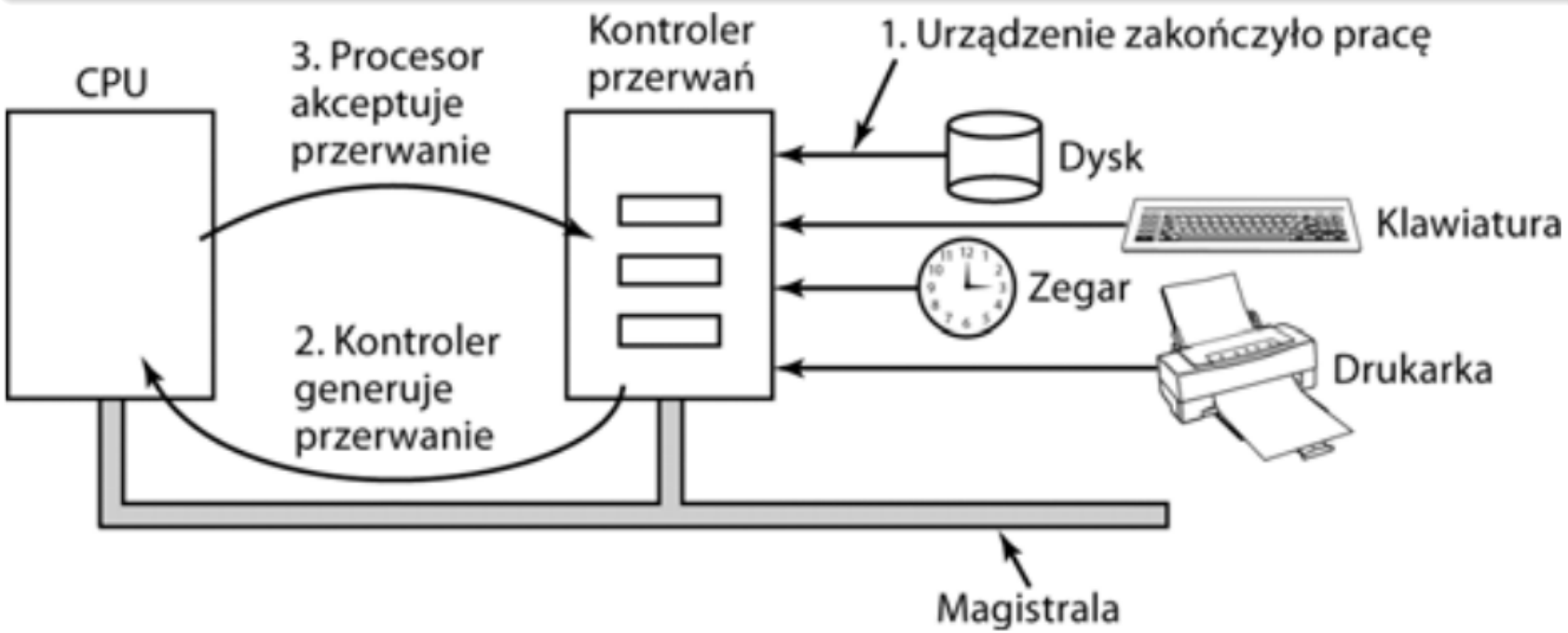
Bezpośredni dostęp do pamięci

Zapis/odczyt pomiędzy pamięcią a urządzeniem zajmuje CPU, alternatywą jest DMA.



Przerwania

Jak wygląda obsługa przerwań (sprzętowych)?



Przerwania (2)

- kontroler przerwań
- wektory przerwań
- procedura obsługi przerwania
- priorytet przerwania
- wyłączanie przerwań

Przerwania (2)

- kontroler przerwań
- wektory przerwań
- procedura obsługi przerwania
- priorytet przerwania
- wyłączanie przerwań

Przerwania (2)

- kontroler przerwań
- wektory przerwań
- procedura obsługi przerwania
- priorytet przerwania
- wyłączanie przerwań

Przerwania (2)

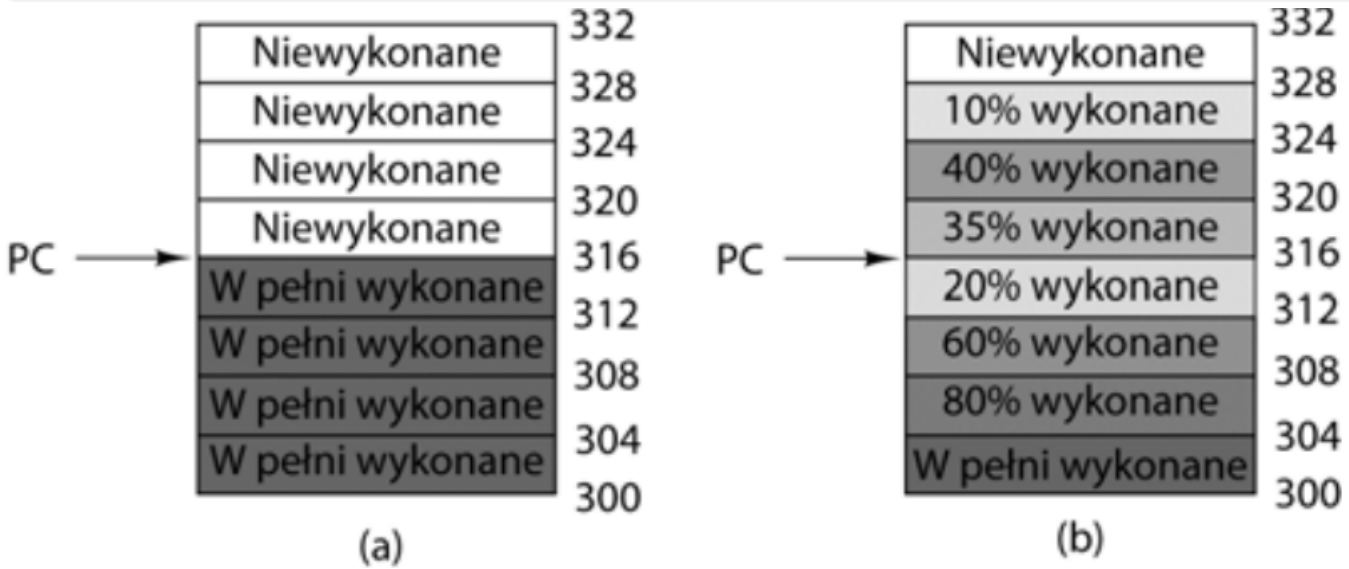
- kontroler przerwań
- wektory przerwań
- procedura obsługi przerwania
- priorytet przerwania
- wyłączanie przerwań

Przerwania (2)

- kontroler przerwań
- wektory przerwań
- procedura obsługi przerwania
- priorytet przerwania
- wyłączanie przerwań

Przerwania

Przerwania precyzyjne i nieprecyzyjne.



Obsługa synchroniczna i asynchroniczna

- synchroniczne (blokujące)
- asynchroniczne (sterowane przerwaniami)

Obsługa synchroniczna i asynchroniczna

- synchroniczne (blokujące)
- asynchroniczne (sterowane przerwaniami)

Rodzaje obsługi wejścia/wyjścia

- programowane
- sterowane przerwaniami
- z wykorzystaniem DMA

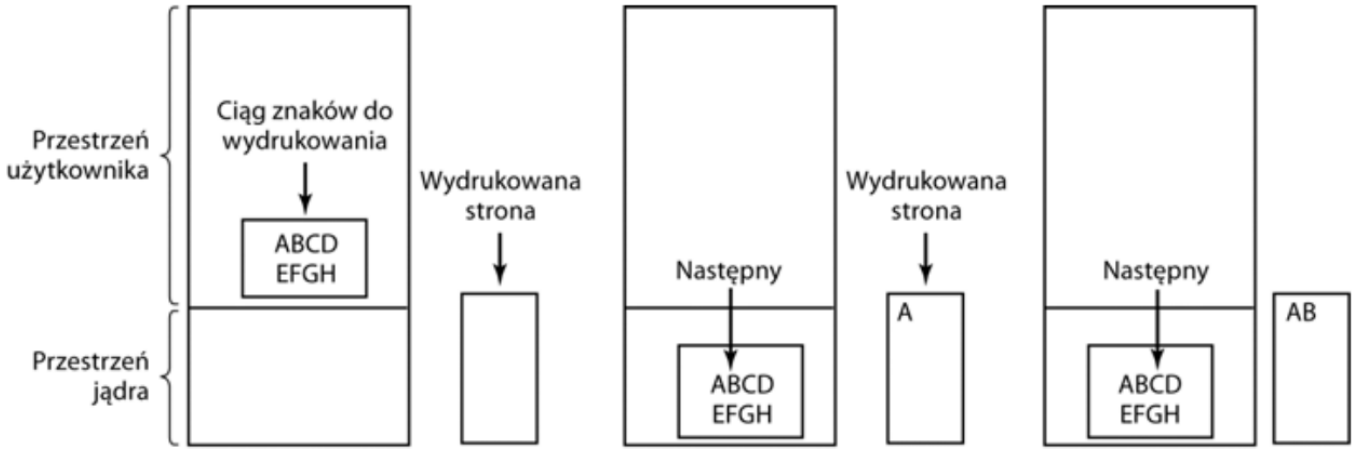
Rodzaje obsługi wejścia/wyjścia

- programowane
- sterowane przerwaniami
- z wykorzystaniem DMA

Rodzaje obsługi wejścia/wyjścia

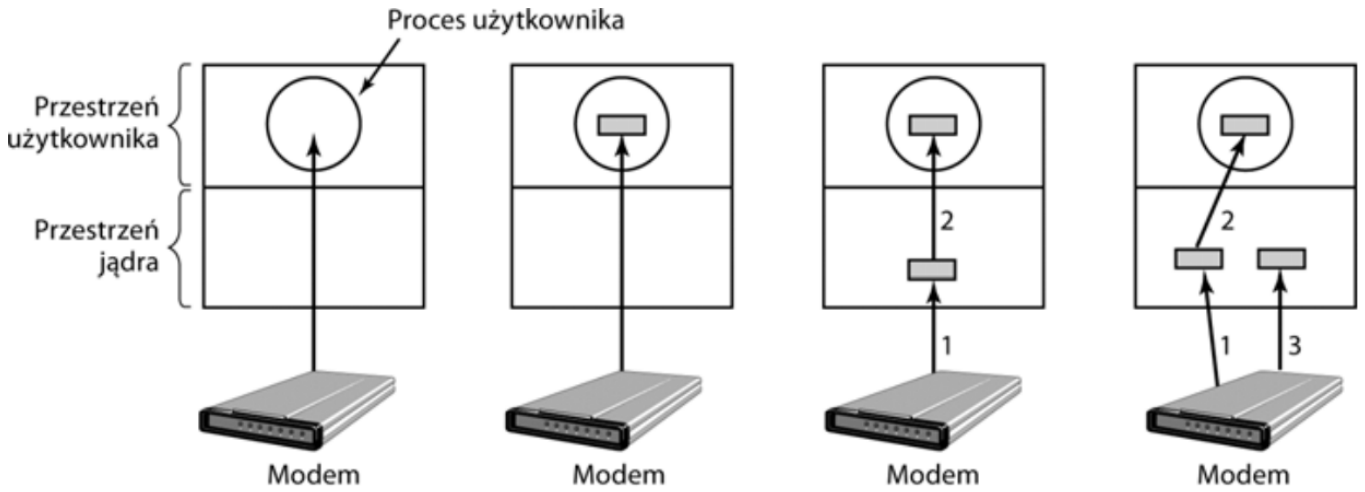
- programowane
- sterowane przerwaniami
- z wykorzystaniem DMA

Programowane wejście/wyjście

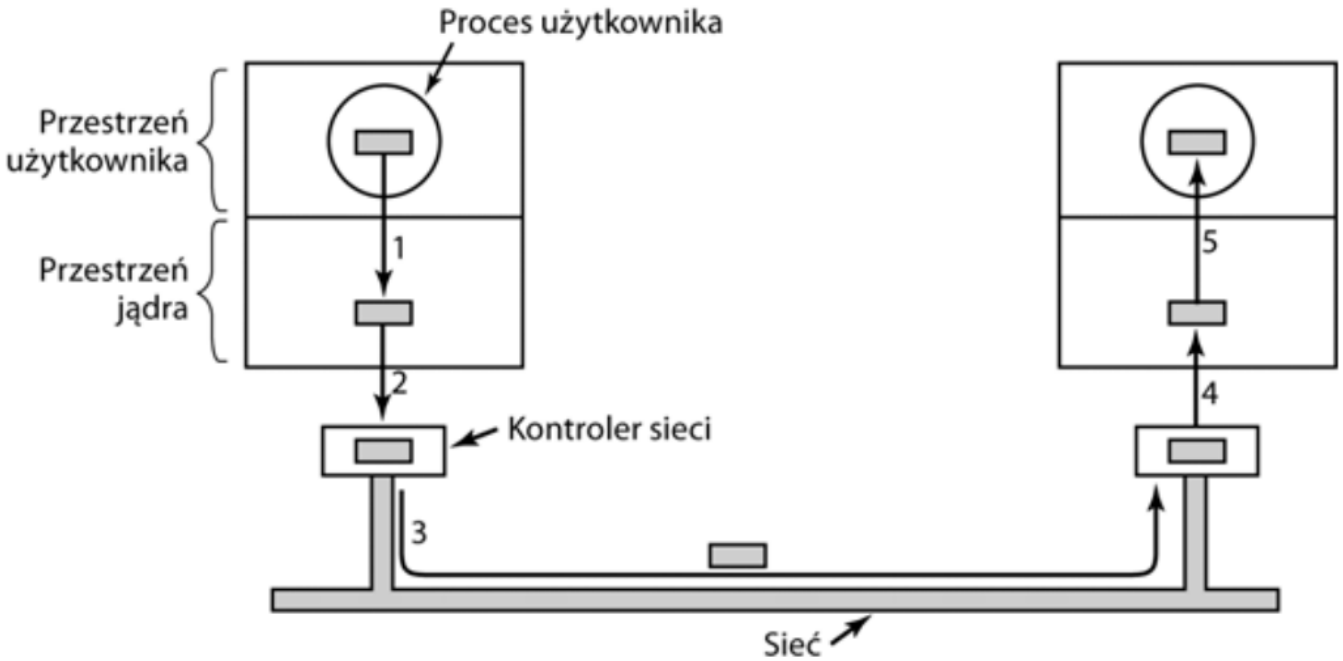



```
copy_from_user(buffer, p, count);              /* p to bufor jądra */
for (i = 0; i < count; i++) {                  /* pętla dla każdego znaku */
    while (*printer_status_reg != READY) ;     /* pętla do czasu uzyskania gotowości */
    *printer_data_register = p[i];             /* wyświetlenie jednego znaku */
}
return_to_user( );
```

Buforowanie



Buforowanie (2)



Obsługa błędów IO

- zwrócenie kodu błędu (np. brak pliku, katalogu)
- zakończenie działania (np. krytyczny błąd sprzętowy)

Obsługa IO – niski/wyższy poziom

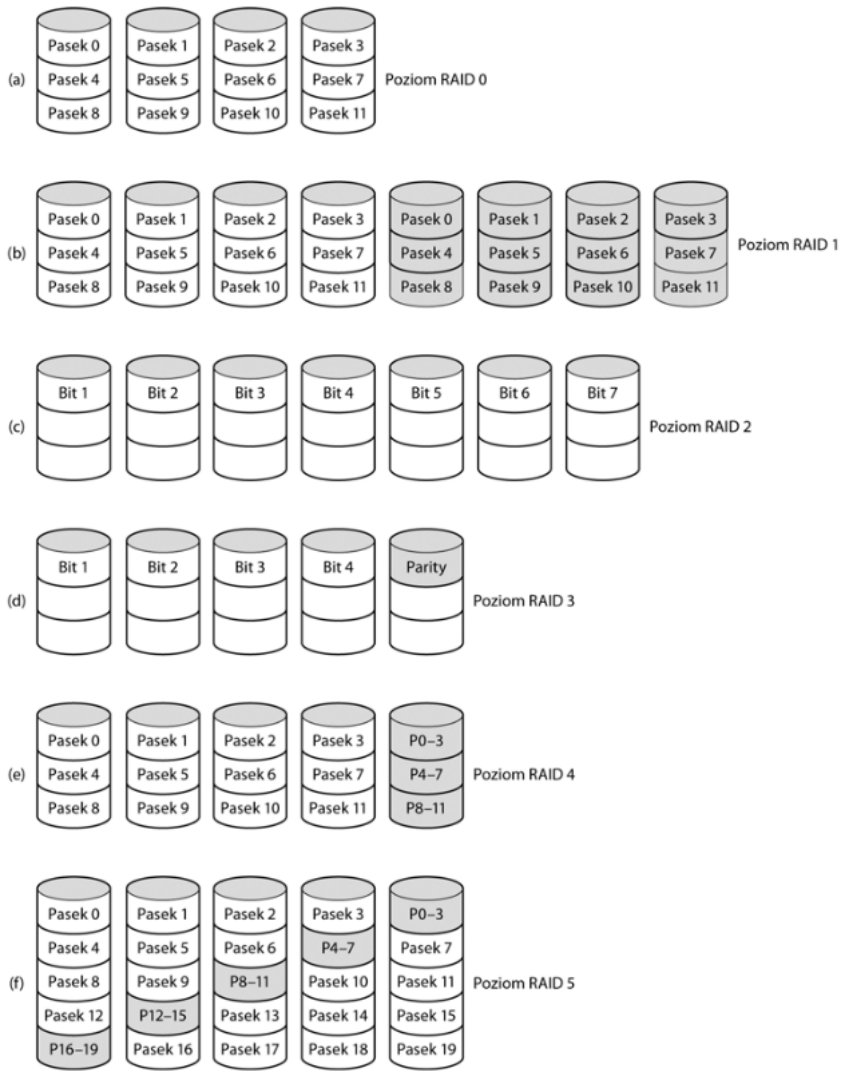
Niski poziom

```
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);
```

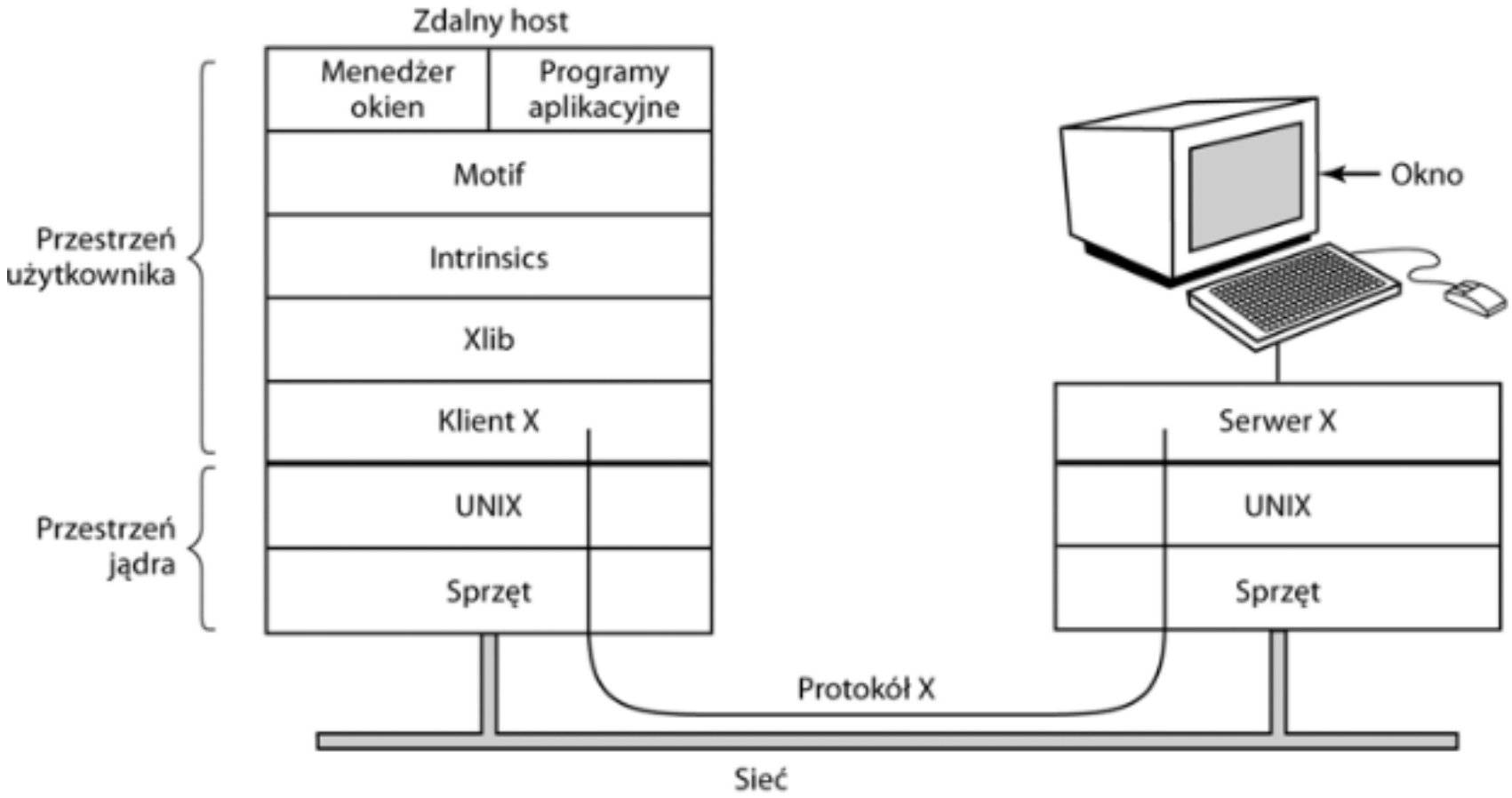
Wyższy poziom – formatowanie, buforowanie

```
#include <stdio.h>

int printf(const char *format, ...);
```

Poziomy RAID



Architektura klient-serwer w X Serverze



Definicja zakleszczenia

Zakleszczenie

W przypadku zbioru procesów do zakleszczenia dochodzi wtedy, gdy każdy proces w zbiorze oczekuje na zdarzenie, które może spowodować tylko inny proces z tego zbioru.

Warunki zakleszczenia

1. wzajemne wykluczenie (przypisany do dokładnie jednego albo dostępny)
2. wstrzymanie i oczekiwanie (p. mogą żądać kolejnych zasobów)
3. brak wywłaszczania (p. musi dobrowolnie zwolnić)
4. cykliczne oczekiwanie (np. dwa p. czekają na wzajemnie zajęte zasoby)

Warunki zakleszczenia

1. wzajemne wykluczenie (przypisany do dokładnie jednego albo dostępny)
2. wstrzymanie i oczekiwanie (p. mogą żądać kolejnych zasobów)
3. brak wywłaszczania (p. musi dobrowolnie zwolnić)
4. cykliczne oczekiwanie (np. dwa p. czekają na wzajemnie zajęte zasoby)

Warunki zakleszczenia

1. wzajemne wykluczenie (przypisany do dokładnie jednego albo dostępny)
2. wstrzymanie i oczekiwanie (p. mogą żądać kolejnych zasobów)
3. brak wywłaszczania (p. musi dobrowolnie zwolnić)
4. cykliczne oczekiwanie (np. dwa p. czekają na wzajemnie zajęte zasoby)

Warunki zakleszczenia

1. wzajemne wykluczenie (przypisany do dokładnie jednego albo dostępny)
2. wstrzymanie i oczekiwanie (p. mogą żądać kolejnych zasobów)
3. brak wywłaszczania (p. musi dobrowolnie zwolnić)
4. cykliczne oczekiwanie (np. dwa p. czekają na wzajemnie zajęte zasoby)

```
typedef int semaphore;
    semaphore resource_1;
    semaphore resource_2;

    void process_A(void) {
        down(&resource_1);
        down(&resource_2);
        use_both_resources( );
        up(&resource_2);
        up(&resource_1);
    }

   void process_B(void) {
        down(&resource_1);
        down(&resource_2);
        use_both_resources( );
        up(&resource_2);
        up(&resource_1);
    }
```

```
semaphore resource_1;
semaphore resource_2;

void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources( );
    up(&resource_2);
    up(&resource_1);
}

void process_B(void) {
    down(&resource_2);
    down(&resource_1);
    use_both_resources( );
    up(&resource_1);
    up(&resource_2);
}
```

A
Żąda zasobu R
Żąda zasobu S
Zwalnia zasób R
Zwalnia zasób S

(a)

B
Żąda zasobu S
Żąda zasobu T
Zwalnia zasób S
Zwalnia zasób T

(b)

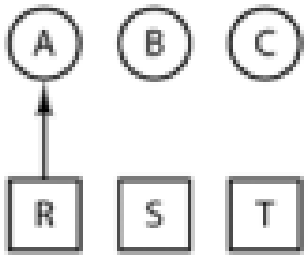
C
Żąda zasobu T
Żąda zasobu R
Zwalnia zasób T
Zwalnia zasób R

(c)

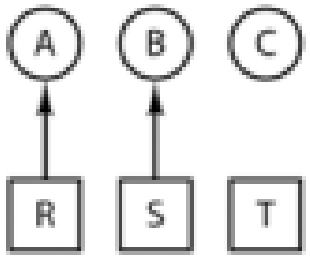
1. Proces A żąda zasobu R
2. Proces B żąda zasobu S
3. Proces C żąda zasobu T
4. Proces A żąda zasobu S
5. Proces B żąda zasobu T
6. Proces C żąda zasobu R

zakleszczenie

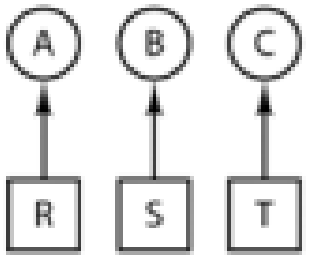
(d)



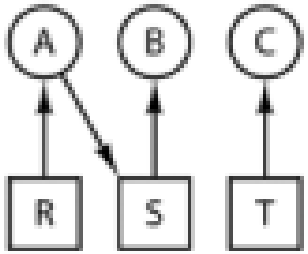
(e)



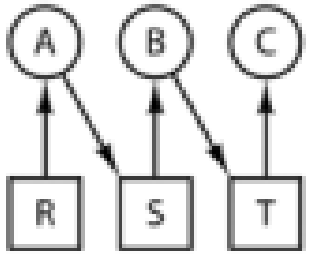
(f)



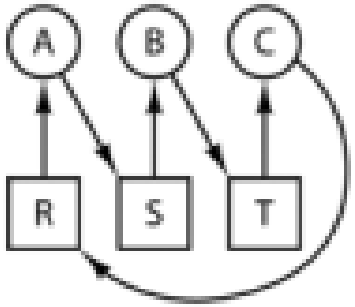
(g)



(h)



(i)



Sposoby radzenia sobie z zakleszczeniami

- ignorowanie problemu (*ostrich algorithm*)
- wykrywanie i podejmowanie czynności zaradczych (po zakleszczeniu)
- dynamiczne unikanie (ostrożna alokacja zasobów)
- prewencja, poprzez negację jednego z czterech wymaganych warunków

Algorytm bankiera

	Posiada	Maks.
A	0	6
B	0	5
C	0	4
D	0	7

Wolne: 10

(a)

	Posiada	Maks.
A	1	6
B	1	5
C	2	4
D	4	7

Wolne: 2

(b)

	Posiada	Maks.
A	1	6
B	2	5
C	2	4
D	4	7

Wolne: 1

(c)