Jądro systemu na podstawie Linuxa

Aleksander Morgała

Programowanie niskopoziomowe

Plan prezentacji:

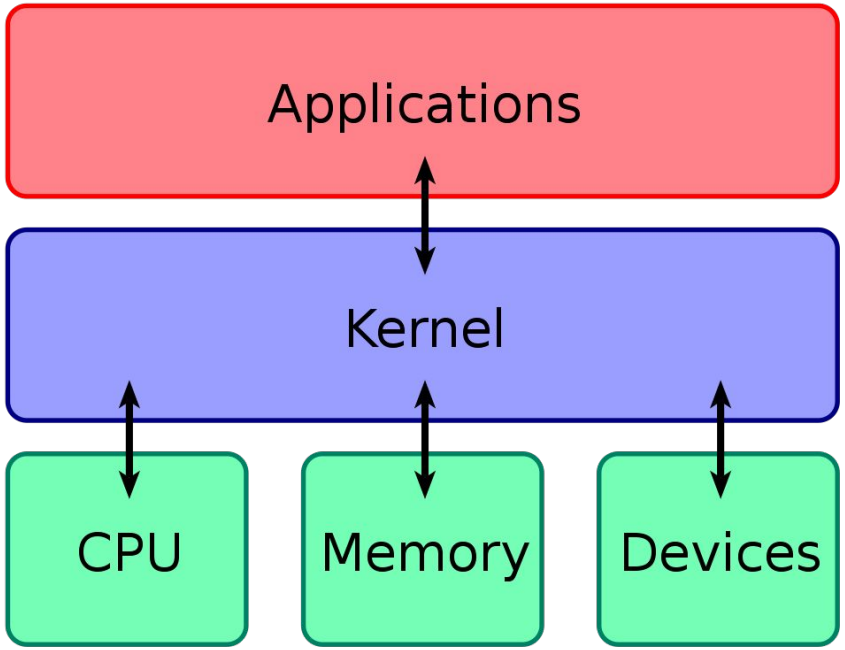
1. Szybkie wprowadzenie do jądra systemu
2. Zarządzanie procesami
3. Trochę o kompilacji jądra i modułach

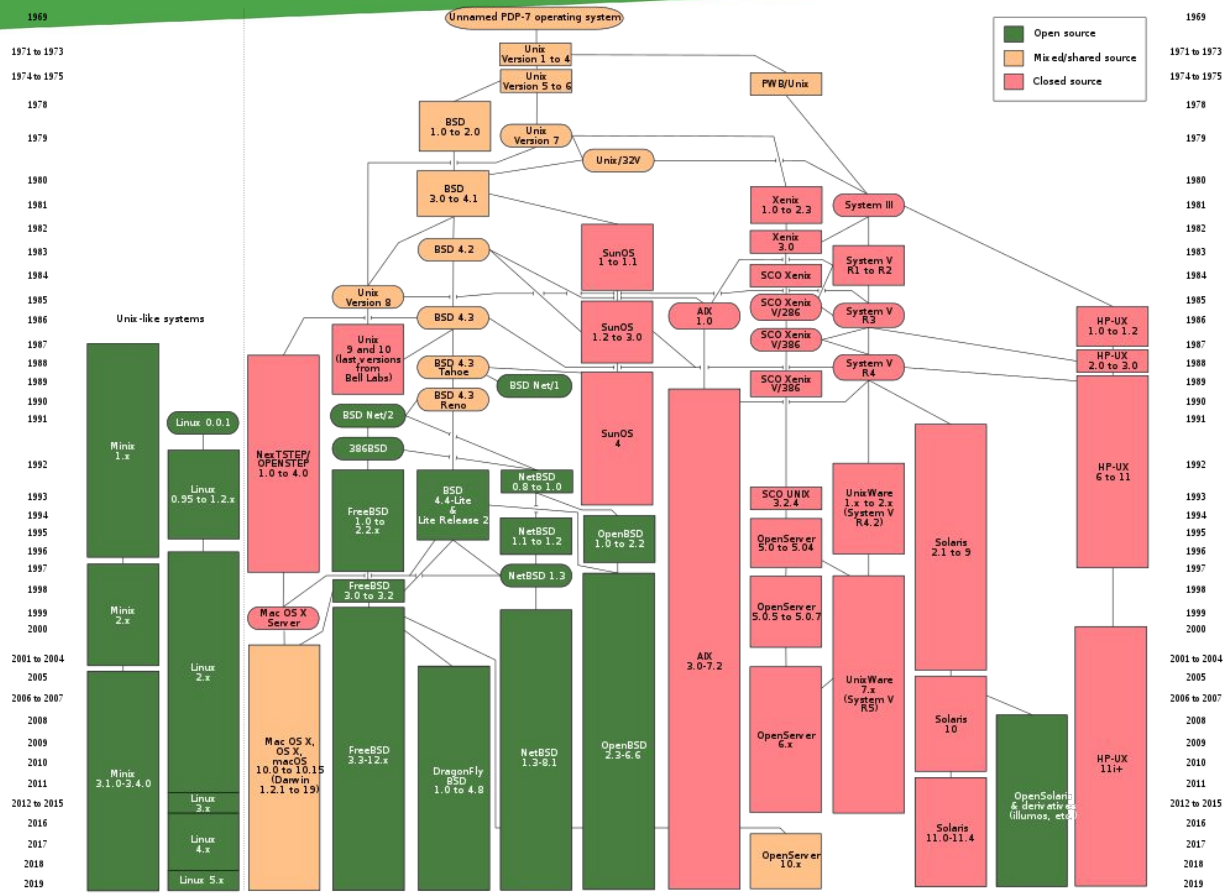
Wprowadzenie do jądra systemu

Szybkie wprowadzenie do jądra systemu

- Co nazywamy jądrem systemu
- Jakie są funkcjonalności jądra
- Rodzaje jąder

Co nazywamy jądrem systemu?





AGH

Programowanie niskopoziomowe

Cechy jądra(linux)

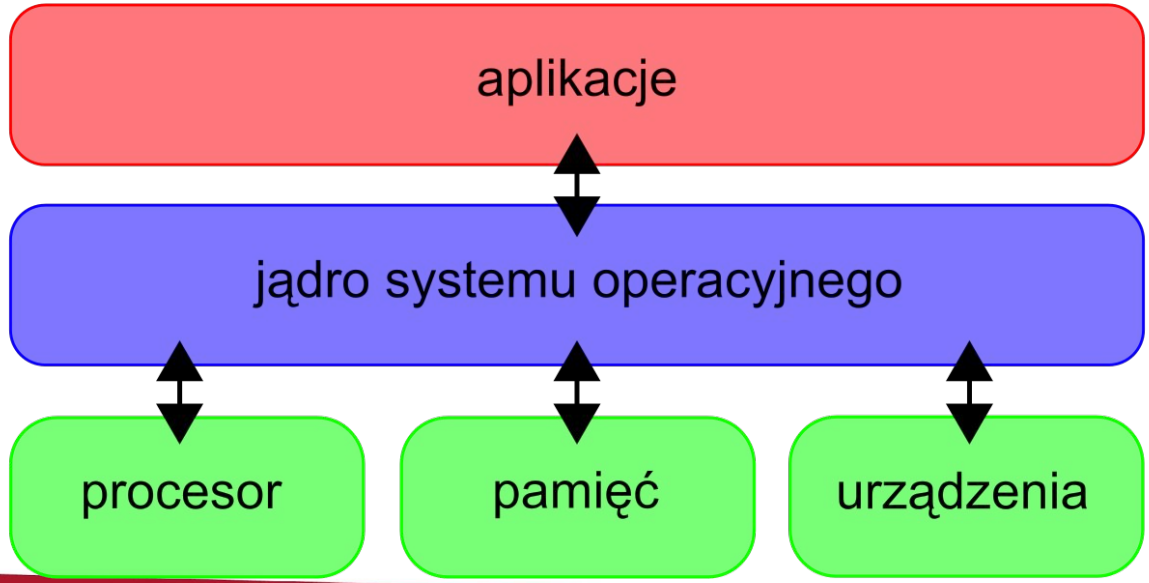
- wieloprocesowość
- wielowątkowość
- wielobieżność
- skalowalność
- wywłaszczalność

Funkcjonalności jądra

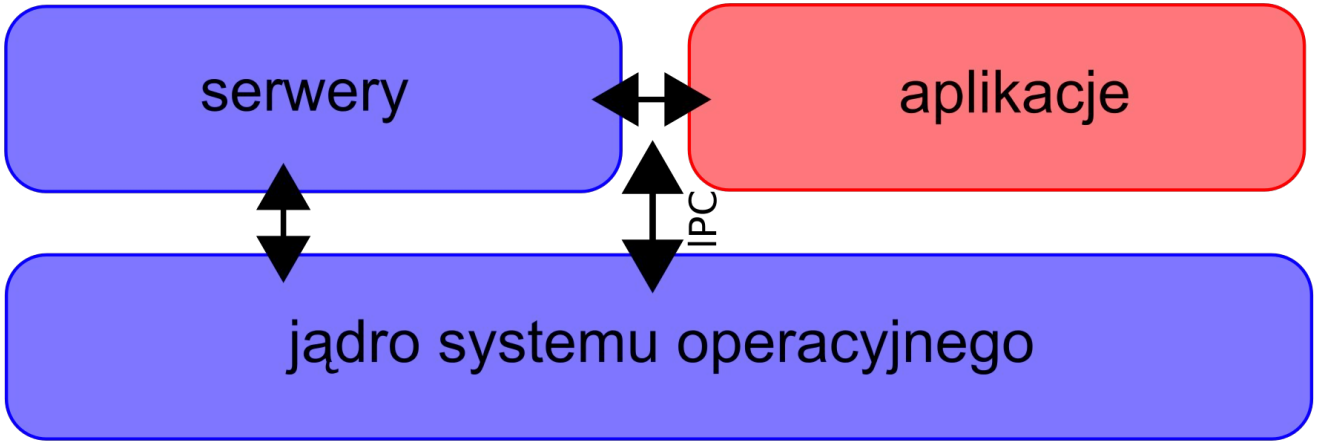
- Zarządzanie pamięcią
- Zarządzanie plikami
- Zarządzanie procesami
- Obsługa urządzeń
- Obsługa połączeń sieciowych

Rodzaje jąder

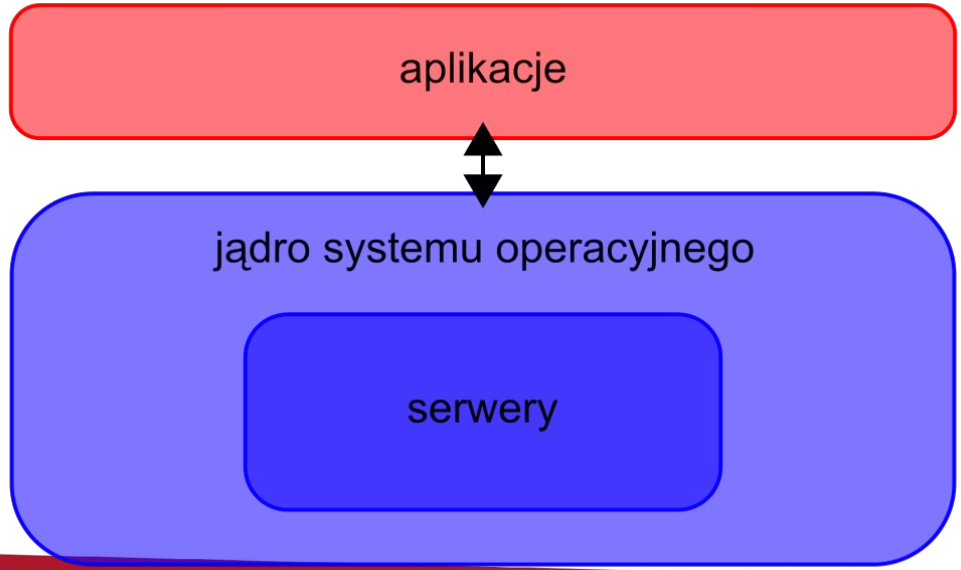
1. Monolitowe - Jądro wszystkie zadania wykonuje bezpośrednio.
   np. Linux, FreeBSD



2. Mikrojądro - Jądro korzysta z “serwerów” do komunikacji z hardwarem.



3. Jądro hybrydowe - Połączenie jądra monolithowego z mikrojądrem. W jądro są wbudowane serwery.



Zarządzanie procesami

a) priorytety i przydział procesora

b) typy procesów (Kernel process/user process)

c) przekazywanie danych pomiędzy procesami

d) planer

e) startowanie, kopiowanie, usuwanie procesów

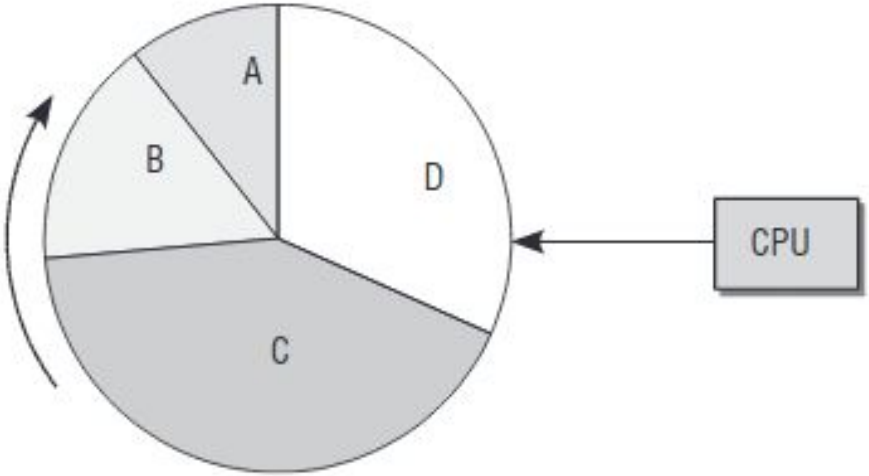
Identyfikacja procesów

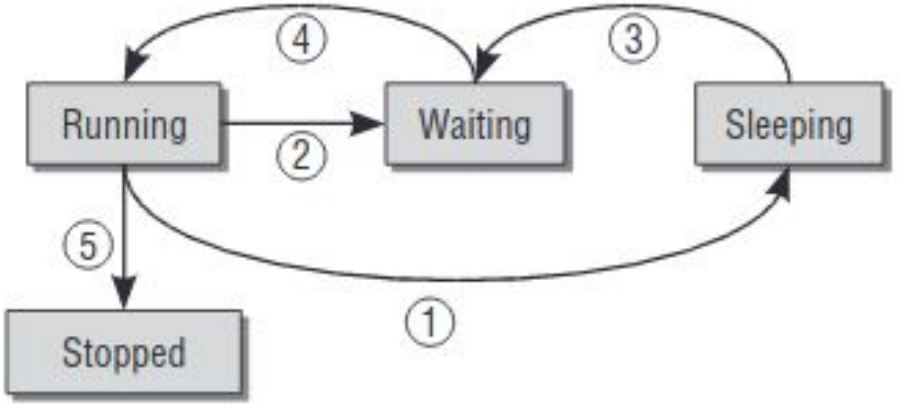
PID - Process Identification Number

TGID - Thread Group ID

PGID - Process Group ID

SID - Session ID


A
B
D
C
CPU


4
3
Running
Waiting
Sleeping
2
5
1
Stopped

Klasyfikacja procesów ze względu na potrzeby czasowe

- Hard real-time process - Procesy z potrzebą wykonania w przeciągu danego czasu, zazwyczaj jak najszybciej. (nieobsługiwane przez kernel linuxa)

- Soft real-time process - Proces z potrzebą wykonania w przeciągu danego czasu, ale dopuszczalne są opóźnienia.

- Normal process - Zwykły proces użytkownika, w naszych systemach korzystamy głównie z takich, wykonuje się kiedy dostanie przydział CPU.

Priorytety

```
ps -e -o uid,pid,ppid,pri,ni,cmd
```

Tryby jądra(wykonywania procesów)

- Zwykły - tryb w którym wykonywane są zwykłe procesy, mogą być wywłaszczone przez inne procesy, system call i przerwania.

- Kernel - tryb w którym przetwarzane są system call. Może być wywłaszczony jedynie przez przerwanie.

Komunikacja międzyprocesowa(IPC)

Używamy gdy:

- Dwa(lub więcej) procesy dzielą dane
- Proces A czeka na wykonanie procesu B
- Proces A przekazuje dane procesowi B

Metody komunikacji pochodzą z System V

Semafory

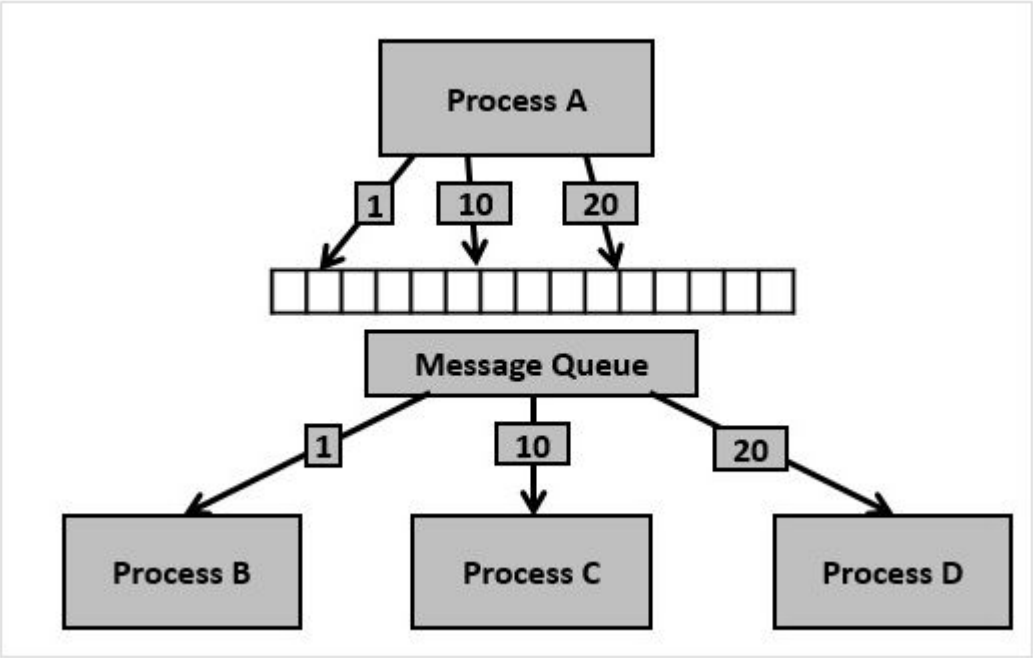
Najprostsza metoda uzgadniania dostępu do danych. Kilka procesów dzieli wspólną zmienną. Arbitralnie ustawiamy klucz semafora(wspólny dla wszystkich procesów)

Dostęp do współdzielonych danych:

1. Ustawienie semafora o podanym kluczu
2. Przy dostępie do danych obniża wartość semafora o 1 i uzyskuje dostęp do danych, lub jeśli semafor już posiada wartość 0, proces “zasypia” i czeka, aż będzie mógł uzyskać dostęp do danych.
3. Kiedy kończy korzystać z danych, zwiększa wartość semafora o 1 i wybudza pierwszy proces w kolejce.

Przykład

Kolejki wiadomości



Wysłanie wiadomości kolejką:

1. Wygenerowanie klucza kolejki
2. Stworzenie lub pobranie numeru kolejki z pomocą klucza z punktu 1.
3. Ustawienia w strukturze numeru wiadomości i wiadomości
4. Wysłanie wiadomości

Odebranie wiadomości z kolejki:

1. Punktu 1,2 z wysyłki.
2. Przeszukanie kolejki w ramach znalezienia wiadomości o wybranym numerze.

   UWAGA: Wiadomość dotrze jedynie do pierwszego procesu który ją odbierze z kolejki!

Przykład

Sygnały

Komunikacja za pomocą sygnałów następuje za pomocą wysyłania sygnałów i ich obsługę przez handlery.

Przykład

Planer

Program zarządzający przełączaniem procesów wykonywanych na CPU. Jego głównym zadaniem jest sprawiedliwe rozdzielenie zasobów pomiędzy procesami z uwzględnieniem ich priorytetów. Zawsze najpierw rozpatrujemy procesy real-time, potem zaczynamy rozpatrywać procesy normalne.

Planer w linuxie obecnie korzysta z dwóch polityk planowania: Real-Time Scheduler i Completely Fair Scheduler.

Problemy które musi rozwiązywać planer

- Unikanie zbyt częstego przełączania procesów
- Unikanie zbyt rzadkiego przełączania procesów
- “Równy” podział czasu procesora
- Obsługa wielu rdzeni

Działanie planera w linuxie

- Wszystkie procesy są przetrzymywane w drzewie czerwono-czarnym, używając przydzielonego czasu procesora jako index.
- Każdy proces ma przypisany maksymalny czas wykonania
- Gdy planer musi wywołać proces to wywołuje proces z najmniejszym indexem
- W zależności od priorytetu procesu różnie naliczany mu jest czas procesora. Np. proces z priorytetem 5 za każdą nanosekundę ma naliczane 3 nanosekundy.

Tworzenie nowych procesów

Fork - “kopiowanie” procesu, kopiujemy całą instancję procesu, razem z pamięcią, co jest wolne i zasobożerne. Aby tego uniknąć linux używa techniki “Copy On Write”.

“Copy On Write” zamiast kopiować całe strony pamięci udostępnia procesów “read-only” dostęp do danych pierwotnego procesu. W momencie próby nadpisania danych przez którykolwiek z procesów proces ten “kopiuje” nadpisane dane i zmienia ich adres w tablicy pamięci stronnicowej.

Dokumentacja

Tworzenie nowych procesów

vfork - skopiowanie procesu, ale z dostępem do tych samych danych co oryginalny proces.

Dokumentacja

Tworzenie nowych procesów

Clone - Klonowanie z wyborem które dokładnie dane chcemy skopiować.

Dokumentacja

Tworzenie nowych procesów

Exec - Zastępuje wywołujący go proces innym procesem podanym jako argument.

Dokumentacja

“Zabijanie” procesów

kill(PID, SIGTERM / SIGKILL)

/usr/include/signal.h

Reprezentacja procesu w C

usr/include/sched.h

Kompilacja

Po co kompilować jądro?

a) Kompilujemy jądro pod nasz procesor, w teorii powinno to zwiększyć wydajność systemu.

b) Dostosowanie jądra do naszych potrzeb. Włączenie/wyłączenie modułów lub opcji.

c) Poznanie możliwości jądra i przetestowanie nowych możliwości.

d) Instalacja innej wersji jądra niż obecnie zainstalowana

Moduły

Moduły to dynamicznie podłączane elementy jądra(głównie sterowniki).

lsmod - listening modułów

modprobe - zarządzanie modułami(przyłączanie/odłączanie)

modinfo <nazwa> - informację na temat modułu

Moduły pozwalają na zaoszczędzeniu pamięci zajmowanej przez jądro, ładowanie jedynie potrzebnych elementów oraz ograniczanie wielkości jądra. Przy instalacji systemu, instalator wykrywa nasz sprzęt

i instaluje automatycznie potrzebne moduły.

Kompilacja jądra

1. Pobranie kodu źródłowego jądra: https://www.kernel.org/
2. Wypakowanie archiwum
3. Konfiguracja pliku .config - *make menuconfig*
4. Instalacja potrzebnych programów -
apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison
5. Kompilacja jądra - *make* (może zajęć parę godzin)
6. Instalacja modułów - *make modules_install*
7. Instalacja jądra - *make install*
8. *update-initramfs*

Źródła:

1 - Polski i Angielski artykuł o jądrze systemu na wikipedii(głównie rysunki i schematy)
2 - “Professional Linux Kernel Architecture” - Wolfgang Mauerer
3 - https://www.tutorialspoint.com/ipc-using-message-queues
4 - https://kernel.org

Dziękuję za uwagę