AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Systemy operacyjne
Wykład 02

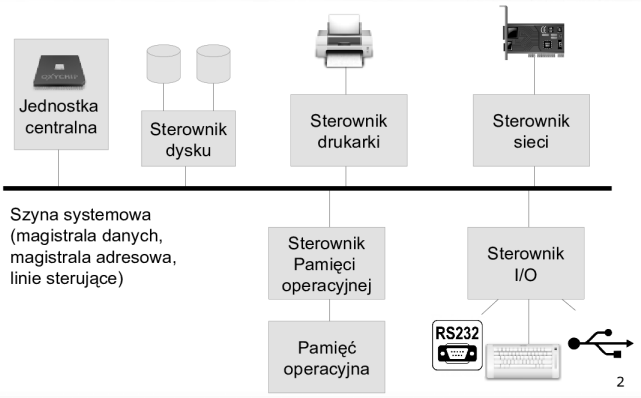
Wersja 2026

dr inż. Marek Wilkus **http://home.agh.edu.pl/~mwilkus**
Wydział Inżynierii Metali i Informatyki Przemysłowej
AGH Kraków

Na podstawie programu opracowanego przez dr inż. Krzysztofa Wilka

AGH

Budowa systemu komputerowego



AGH

Przerwania

- **Przerwanie** jest sygnałem pochodzącym od sprzętu lub oprogramowania, sygnalizującym wystąpienie **zdarzenia**.
- Sygnał przerwania sprzętowego pochodzi z zewnętrznych układów obsługujących przerwania sprzętowe. Służą one do komunikacji z urządzeniami zewnętrznymi (np. napędy dyskowe, sieć, porty).
- Sygnały przerwań od sprzętu wysyłane są do procesora najczęściej za pośrednictwem szyny systemowej.

AGH

Przerwania

- Urządzenie wymagające obsługi procesora (np. kontroler I/O, moduł DMA) posiada specjalne fizyczne połączenie do procesora. (tzw. linia przerwania). W momencie, gdy urządzenie sygnalizuje przerwanie, wystawia odpowiedni sygnał na linii przerwania (z reguły stan wysoki lub zbocze z 0 na 1).

AGH

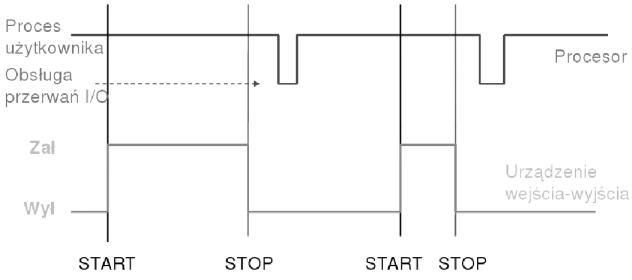
Zdarzenia powodujące przerwanie

- Zakończenie operacji wejścia/wyjścia,
- Niektóre błędy (np. dzielenie przez zero),
- Niedozwolony dostęp do pamięci,
- Zapotrzebowanie na usługę systemu,
- Dostęp przez interfejs zarządzania.
- itd...

Każdemu przerwaniu odpowiada procedura jego obsługi.

AGH

Przerwania procesu wykonującego operację I/O



Wektor przerwań

- Aby przyspieszyć operację obsługi przerwań stosuje się tablicę wskaźników do procedur obsługujących poszczególne przerwania.
- Indeksy tej tablicy odpowiadają numerom urządzeń „generujących” przerwania, a elementami tablicy są adresy procedur obsługujących przerwania (ISR – Interrupt Service Routine).

Wejście w ISR

- Przy przejściu do obsługi przerwania należy zapamiętać adres przerwanego rozkazu, a także np. zawartości rejestrów, jeżeli obsługa przerwania zmienia je.
- W nowych systemach adres powrotny przechowywany jest na stosie systemowym.
- Podczas obsługi jednego przerwania inne są wyłączone, lub ustalone są **priorytety przerwań** (przerwania maskowane).
- Po powrocie z procedury przerwania często nie mamy żadnej informacji że cokolwiek takiego się odbyło.

SMI i NMI

- **NMI** jest niemaskowalne (zawsze zostanie wykonane) i często wykorzystywane jest w obsłudze poważnych problemów (np. zatrzymanie systemu zasygnalizowane przez watchdog timer).
- **SMI** - System Management Interrupt – Jądro systemu zostaje przerwane i nie wiemy co się stało. System coś robił. Uznane przez ITL za zagrożenie bezpieczeństwa, bo wymyka się wszelkim analizatorom. Musi się je wykrywać specjalnymi timerami.
  - Aktualnie (2025-6) zastępowane koprocesorami, czyli już praktycznie niewykrywalne.

Przerwania w systemie Linux

- Przerwań jest dużo, pracy jest dużo, a nie możemy „zakorkować” systemu procedurami obsługi.
- **SoftIRQ** umożliwiają kolejkowanie prac wyzwolonych przez sprzętowe przerwanie (najwyższy priorytet). Inne procesory mogą zająć się wtedy przetwarzaniem zadań, a sam kod przerwania sprzętowego służy wyłącznie "zamówieniu" zadań w softirq wyzwalanym jako zadanie o dużym priorytecie.
- **Tasklets** są niepodzielnymi zadaniami bez własnego stosu/kontekstu, które umożliwiają wykonanie większych prac związanych z soft IRQ, są one z reguły przywiązane do konkretnego rdzenia.

Wyjątki

- Przerwania wewnętrzne, nazywane wyjątkami (ang. exceptions), zgłaszane są przez procesor dla sygnalizowania sytuacji wyjątkowych (np. błąd dzielenia przez 0, nieprawidłowy dostęp do pamięci, naruszenie zasobów współdzielonych).

Podział wyjątków

- **Faults** (niepowodzenia) – sytuacje, w których aktualnie wykonywana instrukcja powoduje błąd. Powrót do wykonywania przerwanego kodu powoduje ponowne wykonanie tej samej instrukcji.
  - Błędy arytmetyczne,
  - Niewłaściwy kod instrukcji (Intel),
  - Operacja na FPU bez FPU,
  - Niewłaściwy dostęp do pamięci,
  - Odwołanie do ramki, której nie ma (trzeba ją sprowadzić).

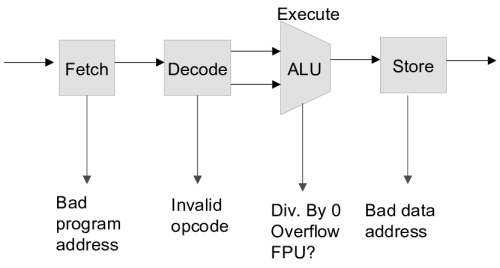
Podział wyjątków (c.d.)

- **Traps** (pułapki) – sytuacja ma na celu intencjonalne wykonanie określonego kodu. Wykorzystywana przede wszystkim w debuggerach gdy niezbędne jest zatrzymanie programu w danym punkcie. Powrót do wykonywania przerwanego kodu powoduje wykonanie **następnej** instrukcji.
  - Niewłaściwy kod instrukcji (Motorola),
  - Specyficzna instrukcja programu,
  - Breakpoint.

Podział wyjątków (c.d.)

- **Aborts** (problemy nienaprawialne) – Wywołane przez błędy, których nie można naprawić. Często nie można określić z którego miejsca w pamięci nastąpiło wywołanie wyjątku ani ponownie wykonać problematycznej instrukcji. Mogą być spowodowane przez błędy sprzętowe, niespójności w systemowych częściach pamięci, błędy niskopoziomowe, "bałagan" wprowadzony w system management mode.
  - Błąd CPU wykryty przez sam procesor,
  - Wyjątek podczas procedury obsługi wyjątku (double fault),
  - Wyjątek podczas obsługi double fault.

Przykłady generowanych wyjątków

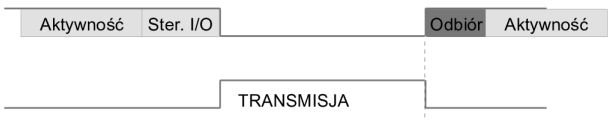


Przerwania programowe

- Z kodu programu wywoływana jest procedura obsługi przerwania;
- Najczęściej wykorzystywane do komunikacji z systemem operacyjnym, który w procedurze obsługi przerwania (np. w DOS 21h, Windows 2Fh, Linux x86 przerwanie 80h) umieszcza kod wywołujący odpowiednie funkcje systemowe w zależności od zawartości rejestrów ustawionych przez program wywołujący, lub do komunikacji z oprogramowaniem wbudowanym (procedury BIOS, firmware).
- Operacje wejścia-wyjścia z reguły nie są dostępne wprost dla programu użytkownika – należy używać funkcji systemowej.

Wejście-wyjście

- Obsługa synchroniczna:



- Obsługa asynchroniczna:

Asynchroniczne I/O

- W czasie wykonywania wejścia-wyjścia jednostka centralna może zostać użyta do obliczeń lub do rozpoczęcia operacji.
- Ponieważ operacje I/O są powolne w porównaniu z szybkością procesora, procesor może w międzyczasie obsłużyć kilka innych zadań.

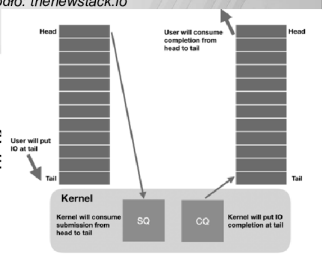
I/O Uring w Linuksie

- Obecne dopiero od Linuksa serii 5.x (2020)
- PROBLEM: Operacje I/O (read, write) są blokujące - program musi czekać na ich zakończenie.
- Wyobraźmy sobie więc serwer bazodanowy oczekujący ze wszystkimi swoimi użytkownikami, aż jeden z nich w końcu uzyska z bazy zamówiony wielki, binarny „blob”, który czyta się z dysku już dłuższy czas…

- Dotychczasowe rozwiązanie: Wątki
  - Problem: Duży nakład na ich tworzenie i zamykanie,
  - Problem: Użycie pamięci,
  - Problem: Brak zapewnienia systemowego buforowania/cache’owania.
  - Narzut na otwieranie i zamykanie deksryptorów plików, nieraz w każdym wątku.

io_uring

Źródło: thenewstack.io



- Program „zamawia” operację I/O dodając ją do końca kolejki SQ (Submission Queue kolejka zgłoszeń) i kontynuuje dalej pracę.
- Program okresowo sprawdza wyniki dla siebie na początku kolejki CQ (Completion Queue, zakonczonych operacji), i przetwarza je odpowiednio.
- Przy operacji na plikach cała kolejka korzysta ze **wspólnego mapowania** deskryptorów (uchwytów) i **wspólnych buforów** jak i możliwe jest **grupowanie** operacji co znacznie zwiększa wydajność.

Co jeszcze można tak robić?

- io_uring_spawn – tworzymy nowe procesy tą metodą.
  + Szybsze – nie trzeba przechodzić zawsze do trybu jądra,
  + Nieblokujące, asynchroniczne,
  + System może to zrobić w wolnej chwili.

  - Program musi wykrywać nie tylko uruchomienie, ale i gotowość procesu potomnego,
  - Kompilatory nadal nie są na to gotowe (jest fork/exec),
  - Co gdy proces się dłużej nie uruchomi z jakiegoś powodu? (brak pamięci, błąd wywołania).

- Aktualnie (2026) eksperymentalna funkcjonalność w jądrze Linux.

Bezpośredni dostęp do pamięci (DMA)

- Gdy urządzenie I/O jest powolne, nie angażuje specjalnie procesora przy przesyłaniu danych bajt po bajcie.
- Dla szybszych urządzeń wygodniej przesyłać cały blok danych (sieć, dyski). Umożliwia to mechanizm Direct Memory Access (DMA) realizowany sprzętowo.

- DMA równocześnie "kradnie" cykle pamięci (procesy wolniej uzyskują dostęp do pamięci!)
  - Priorytety użycia RAM:
    1. Odświeżanie,
    2. DMA,
    3. Procesor,
    4. Opcjonalne bufory i urządzenia.

Tryby pracy procesora

- Tryb użytkownika (z ograniczeniami),
- Tryb monitora / nadzorcy / systemu – wykonuje potencjalne niebezpieczne dla spójności pamięci operacje. Są to tzw. **operacje uprzywilejowane**.

  O ile w trybie użytkownika istnieją pewne, często powiązane z programami, bloki pamięci, w trybie monitora mamy kontrolę nad cała pamięcią, niezależnie od tego co w niej działa.

Tryby pracy procesora

- **Operacje wejścia/wyjścia** nie są bezpośrednio dostępne dla użytkownika (musi o nie prosić system operacyjny).
- Użytkownik ma dostęp do pamięci przydzielonej **tylko swojemu programowi.**
- System nie może utracić kontroli nad procesorem np. przez nieskończoną pętlę w programie użytkownika.

Praca procesorów Intel

- Tryb **rzeczywisty** – Zgodność z podstawowym, 16-bitowym x86 z adresowaniem pamięci włącznie. Używany na wczesnych etapach uruchamiania komputera (UEFI go opuszcza, BIOS rzadko, Win9x – wraz z inicjalizacją VMM, WinNT i Unix – jądra).
- Tryb **chroniony** – działa ochrona pamięci, pamięć wirtualna, kontekst i działania na MMU. Większość systemów operacyjnych działa w tym trybie.
- Tryb **wirtualny** – możliwość uruchomienia wirtualnego CPU w trybie rzeczywistym. Używany w niektórych sterownikach w systemach 32-bit, w systemach 64bit w zasadzie niedostępny.

Proces

- Proces jest programem, który jest aktualnie wykonywany.
- Jest to jednostka pracy w systemie.
- System składa się ze **zbioru procesów**, których część to procesy systemu, pozostałe są procesami użytkowymi.

Zarządzanie procesami przez OS

- Tworzenie i usuwanie procesów użytkowych i systemowych,
- Wstrzymywanie i wznawianie procesów,
- Dostarczanie mechanizmów synchronizacji procesów,
- Dostarczanie mechanizmów komunikacji między procesami.
- Dostarczanie mechanizmów obsługi zakleszczeń.

Zarządzanie pamięcią przez OS

- Ewidencja aktualnie zajętych obszarów pamięci, informacja o użytkownikach danych obszarów,
- Decydowanie o załadowaniu procesów do zwolnionych miejsc w pamięci,
- Przydzielanie i zwalnianie pamięci stosownie do potrzeb,

Zarządzanie plikami przez OS

- Tworzenie i usuwanie plików,
- Tworzenie i usuwanie katalogów,
- Dostarczanie informacji do manipulowania plikami i katalogami, obsługa metadanych systemu plików,
- Odwzorowanie plików na obszary pamięci pomocniczej,
- Składowanie plików na nośnikach w pamięci nieulotnej.

Inne funkcje systemu operacyjnego

- Zarządzanie systemem wejścia/wyjścia (buforowanie, pamięć, spooling, programowanie interfejsów, moduły sterujące),
- Zarządzanie pomocniczą pamięcią dyskową,
- Praca sieciowa,
- System ochrony,
- Uruchomienie systemu interpretacji poleceń (powłoka).

Usługi systemu operacyjnego

- Wykonanie programu,
- Operacje wejścia-wyjścia,
- Manipulowanie systemem plików,
- Komunikacja między procesami,
- Wykrywanie błędów.
- Przydzielanie zasobów,
- Rozliczanie
- Ochrona

Funkcje (wywołania) systemowe

- Tworzą **interfejs pomiędzy wykonywanym programem a systemem operacyjnym.**
- Poprzez funkcje systemowe program użytkownika "daje zlecenia" systemowi operacyjnemu.

Funkcje systemowe – nadzorowanie procesów

- Załadowanie lub wykonanie programu,
- Zakończenie lub zaniechanie procesu,
- Utworzenie lub zakończenie procesów potomnych,
- Pobieranie lub ustawianie parametrów procesów,
- Oczekiwanie "czasowe",
- Oczekiwanie na zdarzenie lub sygnalizacja jego wystąpienia,
- Przydział i zwolnienie pamięci.

Funkcje systemowe – operacje na plikach

- Utworzenie lub usuwanie plików,
- Otwarcie lub zamknięcie plików,
- Zapis, Odczyt lub zmiana położenia w pliku,
- Pobranie lub ustawianie atrybutów i metadanych pliku.

Funkcje systemowe – operacje na urządzeniach

- Zarezerwowanie lub zwolnienie urządzenia,
- Zapis, odczyt lub zmiana położenia (o ile to możliwe),
- Pobieranie i ustawianie atrybutów i ustawień urządzenia,
- Logiczne podłączanie lub odłączanie urządzeń.

Funkcje systemowe – utrzymywanie informacji

- Pobieranie lub ustawianie daty/czasu,
- Pobieranie lub ustawianie danych systemowych, konfiguracji, ustawień NVR (o ile istnieją),
- Pobieranie atrybutów procesów, plików lub urządzeń,
- Ustawianie atrybutów procesów, plików lub urządzeń,

Funkcje systemowe - komunikacja

- Tworzenie lub usuwanie połączeń komunikacyjnych,
- Nadawanie i odbieranie komunikatów,
- Przekazywanie informacji o stanie systemu i komponentów,
- Przyłączanie i odłączanie urządzeń zdalnych.

Przykładowe funkcje systemowe

- Tworzenie i uruchamianie programów:
  - fork(2)
  - execve(2)
  - exec(5)
- Operacje na plikach:
  - creat(2)
  - open(2)
  - read(2)
  - write(2)
  - close(2)
  - chown(2)
  - chmod(2)
- Operacje na katalogach:
  - opendir(3)
  - closedir(3)
  - scandir(3)
- Środowisko:
  - getenv(3)
  - setenv(3)
  - putenv(3)

Programy systemowe

Wchodzą w skład systemu korzystając z funkcji systemowych umożliwiają wypełnianie zadań.

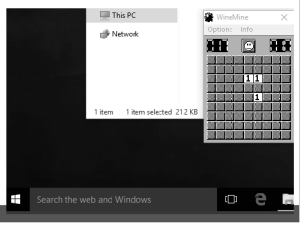
- Manipulowanie plikami,
- Informowanie o stanie systemu,
- Tworzenie i modyfikacja zawartości plików,
- Ładowanie i wykonywanie programów,
- Komunikacja.

API, ABI, zgodność oprogramowania

- Funkcje oferowane przez system dla programów użytkownika tworzą **API** - Application Programming interface.
  - W Linuksie, oferowane funkcje systemowe i GLIBC są standardowe, stosunkowo niewielkie i realizują podstawowe operacje. Zgodność na poziomie API zapewniona jest przez dłuższy czas.
  - W Windows, funkcji API przybywa z wersji na wersję, lecz zachowywane są starsze funkcje celem zgodności wstecznej. Dzięki temu można kompilować wiele programów napisanych na starsze systemy.

ABI

- Gdy program zostaje skompilowany, komunikuje się z systemem za pomocą interfejsu binarnego (**ABI** - Application Binary Interface), - odpowiadają API struktury pamięci i wywoływane bloki kodu.
  - W Linuksie, ABI jest stabilne, lecz nie jest to ścisłe wymaganie. Wiele programów więc w kolejnych wersjach trzeba przekompilować. Nie pomagają również stale ewoluujące biblioteki. Ze względu na to, że większość oprogramowania jest open source, nie stanowi to krytycznego problemu w typowych sytuacjach.
  - W Windows, ABI jest szczególnie stabilne. Dzięki temu programy własnościowe, których kod jest tajny, można uruchamiać a kompatybilność wsteczna jest zachowana.



Źródło: N. Lineback

Jądro Linuksa

- Stałe API na zewnątrz (funkcje systemowe),
- W większości przypadków stałe ABI na zewnątrz.
- Ale wewnątrz samego jądra ani API, ani ABI nie jest zachowywane. Istnieje stabilne API dla sterowników, ale ABI już zmienia się w kolejnych wersjach.

  Co z tym zmiennym ABI mają począć autorzy sterowników?

  DKMS - Dynamic Kernel Module Support
- Gdy instalowane jest nowe jądro systemu, z nim instalowane są jego nagłówki (pliki .h). W systemie jest już kompilator. Dodatkowe sterowniki (np. grafika nVidia) występują w postaci kodu źródłowego i są automatycznie dokompilowane do nowego jądra jako moduły.

ABI we współpracy

- Program współpracujący z biblioteką musi wiedzieć jak wywołać z niej funkcję. Jak przesłać do niej zmienną konkretnego typu. Jak odebrać z niej wynik. To również ABI.
- Kompilatory **nie mogą** naruszać ABI z wersji na wersję. Spowoduje to niekompatybilność binariów w zależności od wersji kompilatora, który je zbudował i konieczność ich przekompilowania.
- Niestety czasami się to dzieje (time_t i problem roku 2038!). Niektóre systemy, w celu zapewnienia zgodności z przestarzałymi programami, których kody źródłowe zaginęły, są wydawane w wersjach intencjonalnie zbudowanych w starych kompilatorach (Haiku OS).

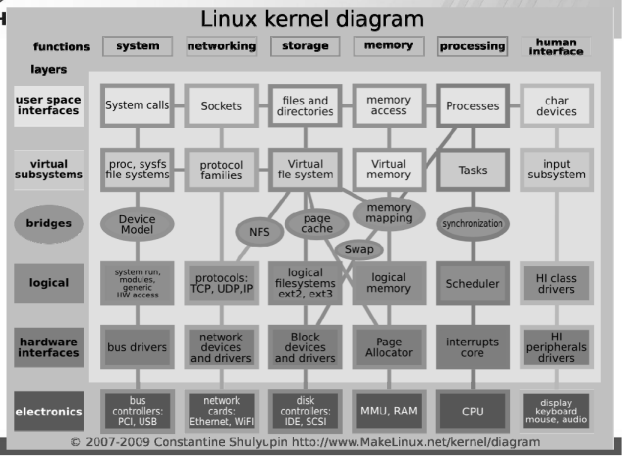
Mac OS

- Toolbox ROM - stopień pośredni między funkcją firmware a funkcją systemową. Wyzwalanie przerwania "A-trap" przy intencjonalnie niewłaściwym rozkazie procesora powoduje wywołanie funkcji z pamięci stałej sprzętu. W ten sposób zaimplementowane są nawet proste operacje graficzne.
- Gdy procesory stały się odpowiednio szybkie, a duża pamięć stała zaczęła generować koszty, symulowano zachowanie Toolbox za pomocą funkcji systemu, a później porzucono to całkowicie.
- Mac OS X ma już zestaw wywołań systemowych w dużej mierze zgodnych z BSD.

Struktura systemu UNIX

Użytkownicy		
Powłoki i polecenia Kompilatory i interpretery Biblioteki systemowe		
Interfejs funkcji systemowych jądra		
Sygnały Obsługa terminali System znakowego wejścia-wyjścia Moduły sterujące terminali	System plików Wymiana System blokowego wejścia-wyjścia Moduły sterujące dysków i taśm	Planowanie przydziału procesor Zstępowanie stron Stronicowanie na żądanie Pamięć wirtualna
Interfejs między jądrem a sprzętem		
Sterowniki terminali Terminale	Sterowniki urządzeń Dyski i taśmy	Sterowniki pamięci Pamięć operacyjna

Linux Kernel
(dla uproszczenia pominięto systemd i PA)



Dziękuję za uwagę