AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Systemy operacyjne
Wykład 09

Wersja 2024

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

System wejścia-wyjścia

AGH

Poziomy abstrakcji dostępu do I/O

- Program użytkownika pisze wprost do pamięci urządzenia lub do jego rejestrów, odpytuje urządzenie, przeprowadza - bardzo niski poziom, programy DOS.
  - Drivery? Jakie drivery? Każdy program ma własne.
- Program użytkownika korzysta z wywołań BIOSu lub procedur w pamięci ROM urządzenia, co może być mniej lub bardziej standardowe - jw.
  - ...ale to działa tylko dla prostych urządzeń.
- Program użytkownika korzysta ze standaryzowanego interfejsu urządzenia odwołując się do jego programu obsługi działającego równolegle w tej samej przestrzeni - np. programy Windows 3.x.
  - Tu wciąż da się zawiesić system z poziomu programu.

AGH

Poziomy abstrakcji dostępu do I/O - c.d.

- Program korzysta ze standardowych funkcji systemu operacyjnego by z jego pomocą ustalić położenie urządzenia oraz jego możliwości. Dalsze operacje może wykonywać przez system (preferowane) lub w niektórych przypadkach samodzielnie. - Windows 9x, nieco wyższy poziom.
- Program do operacji z urządzeniami wykorzystuje wyłącznie API programistyczne systemu. Funkcje są ujednolicone i program może korzystać z niemal dowolnego modelu urządzenia o ile jego driver jest zainstalowany w systemie. - Windows NT, wysoki poziom abstrakcji.
- Jak wyżej, ale dodatkowo urządzenia są dostępne wprost dla użytkownika, np. jako pliki (UNIX - rozszerzenie powyższego)

AGH

System wejścia-wyjścia

- Trzy rodzaje urządzeń I/O:
  - Urządzenia pamięci (dyski, stacje, taśmy),
  - Urządzenia przesyłania danych (karty sieciowe, modemy, porty),
  - Urządzenia komunikacji z człowiekiem (klawiatury, myszy, monitory).
- Niektóre urządzenia mogą przynależeć do kilku kategorii (np. skaner pobiera dane, ale i ma przyciski HID, drukarka dane odbiera, i przekształca na formę zrozumiała dla człowieka).

AGH

Różnice między urządzeniami I/O

- Urządzenie znakowe - przesyła bajty (znaki) z osobna jeden za drugim (terminal).
- Urządzenie blokowe - przesyła jednorazowo całe bloki (dysk).
- Dostęp sekwencyjny - dane przesyłane kolejno w sposób uporządkowany (modem).
- Dostęp swobodny - można mieć dostęp do danych w różnych miejscach, niekoniecznie kolejno (CD-ROM)
- Przesyłanie synchroniczne - taktowane zegarem (taśma).
- przesyłanie asynchroniczne - w nieokreślonych chwilach czasu, sterowane startem i stopem (klawiatura).
- Urządzenie dzielone - przez kilka procesów (dysk).
- Wyłączne - tylko dla jednego użytkownika (taśma).
- Szybkość działania - od B/s do GB/s.
- Kierunek przesyłania - czytanie, pisanie lub czytanie i pisanie.

Sterowanie urządzeniami wejścia-wyjścia

Przekazywanie poleceń z procesora do interfejsu:

Interfejs posiada rejestry do pamiętania danych i sygnałów sterujących. Procesor pisze i czyta do rejestrów w interfejsie.

- Dalej możliwości są dwie:
  - Procesor posiada specjalne rozkazy do pisania i czytania portów,
  - Operacje we-wy odbywają się w pamięci - rejestry są odwzorowywane w przestrzeni adresowej procesora.

W komputerach PC zastosowane są obydwie metody:

- Dla kontrolera grafiki - ekran odwzorowany w pamięci.
- Dla portów szeregowych - rejestry we-wy i bufory.

Port wejścia-wyjścia

- Stan (czytany przez procesor) – np. zakończenie wykonywania polecenia, dostępność danych, błąd urządzenia;
- Sterowanie (zapisywane przez procesor) – np. rozpoczęcie polecenia, zmiana trybu pracy, otwarcie/zamknięcie urządzenia.
- Dane wejściowe (czytane przez procesor) – dane właściwe z urządzenia np. kody klawiszy z klawiatury, bajty z portu komunikacyjnego
- Dane wyjściowe (Zapisywane przez procesor) – dane dla danego urządzenia, np. bit-mapa dokumentu dla drukarki.

Układ FIFO – "magazyn pośredni" – buforuje dane, których komputer lub urządzenie nie może w danej chwili odebrać.

Odpytywanie

- Do uzgadniania pomiędzy procesorem a urządzeniem w prostym schemacie producent-konsument wystarczą dwa bity:
  - od strony procesora bit gotowości polecenia w rejestrze poleceń - sygnalizujący kompletne polecenie dla urządzenia,
  - od strony urządzenia bit zajętości (w rejestrze stanu), sygnalizujący że urządzenie jest zajęte pracą.
- Kolejność działań przy uzgadnianiu:
  - Procesor realizuje aktywne czekanie, dopóki bit zajętości jest ustawiony
  - Procesor ustawia bit pisania i wpisuje bajt danych do rejestru danych wy.
  - Procesor ustawia bit gotowości polecenia
  - Sterownik ustawia bit zajętości po zauważeniu bitu gotowości polecenia
  - Sterownik czyta rejestr poleceń, rozpoznaje polecenie pisania. Czyta bajt danych z rejestru i wykonuje na urządzeniu operację wejścia-wyjścia
  - Sterownik czyści bit gotowości polecenia, bit błędu, a na końcu bit zajętości

I tak dla każdego bajtu danych

Przerwania

Jeśli urządzenie jest rzadko gotowe do działania, odpytywanie staje się nieefektywne (procesor większość czasu poświęca na aktywne czekanie).

Mechanizm przerwań:

- Procesor ma końcówkę (pin) badającą stan linii zgłaszania przerwań po wykonaniu każdego rozkazu.
- Jeśli procesor wykryje wystąpienie przerwania, to wykonuje operacje zachowania stanu bieżącego procesu i przechodzi do procedur obsługi przerwań.
- Po wykonaniu niezbędnych operacji procesor wraca do wykonywania przerwanego zadania
- W złożonych architekturach komputerów możliwa zaawansowana obsługa przerwań:
  - opóźnianie obsługi przerwania podczas działań krytycznych,
  - maskowanie przerwań (dwie linie przerwań - maskowalna i niemaskowalna)
  - przerwania wielopoziomowe o różnym priorytecie.

Bezpośredni dostęp do pamięci

- Jest używany w celu uniknięcia transmisji bajt-po-bajcie (zwanego programowanym wejściem-wyjściem) dla urządzeń transmitujących wielkie ilości danych (np dysk), co oszczędza wiele cykli procesora.
- Wiele procedur związanych z transmisją jest wtedy wykonywana przez specjalizowany procesor – **sterownik bezpośredniego dostępu do pamięci** (DMA controller).
- Przed rozpoczęciem transmisji w trybie DMA, procesor zapisuje w pamięci blok sterujący DMA (wskaźnik do źródła, adres docelowy, liczba bajtów do przesłania), następnie przesyła do sterownika DMA adres tego bloku i przechodzi do wykonywania innych prac.
- Sterownik DMA wykonuje transmisję, przejmując w tym czasie sterownie szyną pamięci. Procesor nie ma wtedy dostępu do pamięci, ale może korzystać z cache i rejestrów.

Wejście-wyjście z blokowaniem

- Blokowanie uwalnia procesor od aktywnego czekania – proces przenoszony jest do kolejki procesów czekających. Po zakończeniu we-wy proces przechodzi do kolejki procesów gotowych.
- Niektóre procesy wymagają wejścia-wyjścia bez blokowania, np proces w którym sygnały z klawiatury lub myszy przeplatają się z przetwarzaniem i wyświetlaniem na ekranie, albo czytanie z dysku z dekompresją danych.
- W aplikacjach wielowątkowych można zablokować pewne wątki, a inne zostawić aktywne.

Podsystem I/O w jądrze systemu

- **Planowanie wejścia-wyjścia** ma na celu poprawę wydajności systemu, polepszenie wspólnego korzystania z urządzeń przez procesy i zmniejszenie średniego czasu oczekiwania.
- **Buforowanie** - dopasowanie prędkości „producenta" i „konsumenta" danych (podwójne buforowanie), dopasowanie urządzeń operujących na różnych wielkościach bloków danych (np pakiety sieci a bloki na dysku)
- **Przechowywanie podręczne** (caching) - zapamiętanie kopii danych w szybkiej pamięci podręcznej
- **Spooling** - użycie bufora do przechowywania danych przeznaczonych dla urządzenia, które nie dopuszcza przeplatania danych z różnych procesów (np drukarka)
- **Obsługa błędów**
- **Struktury danych jądra** - jądro musi przechowywać informacje o stanie używanych składowych wejścia-wyjścia.

Podsystem I/O w jądrze

Podsystem nadzoruje:

- Zarządzanie przestrzenią nazw plików i urządzeń,
- Przebieg dostępu do plików i urządzeń,
- Poprawność formalną operacji,
- Przydzielanie miejsca w systemie plików,
- Przydział urządzeń,
- Buforowanie, caching oraz spooling,
- Planowanie operacji wejścia-wyjścia
- Doglądanie stanu urządzeń, obsługę błędów, czynności naprawcze po awarii.
- Konfigurowanie i wprowadzanie w stan początkowy modułu sterującego.

Poprawianie wydajności I/O

Podsystem ma na celu:

- Zmniejszać liczbę przełączeń kontekstu,
- Zmniejszać liczbę kopiowań danych w pamięci podczas przekazywania od urządzenia do aplikacji,
- Zmniejszać częstość przerwań poprzez przesyłanie dużych porcji informacji, przez optymalizację sterowników i stosowanie odpytywania, gdzie to korzystne,
- Zwiększać współbieżność poprzez stosowanie sterowników pracujących w trybie DMA,
- Realizować elementarne działania za pomocą sprzętu i pozwalać na ich współbieżne wykonywanie w sterownikach,
- Równoważyć wydajność procesora, podsystemów pamięci, szyny i operacji wejścia-wyjścia.

Wydajność pamięci dyskowej (HDD) od czego zależy?

- Czytanie lub pisanie wymaga ustawienia głowicy nad określoną ścieżką i na początku konkretnego sektora. Potrzebny na to jest **czas dostępu** - suma czasu przeszukiwania i opóźnienia obrotowego.
  - Czas przeszukiwania (seek time) - czas potrzebny na ustawienie głowicy nad ścieżką
  - Opóźnienie obrotowe (rotational latency) - czas potrzebny na obrót właściwego sektora pod głowicę,
- **Czas przeszukiwania** ma duży wpływ na wydajność.
- Możemy go optymalizować, ponieważ system operacyjny zarządza kolejką żądań dysku, choć dynamiczna translacja geometrii dysku nie pomaga.

Wydajność pamięci dyskowej (SSD) Od czego zależy?

- Dyski SSD są znacznie szybsze pod względem dostępu od magnetycznych.
- Defragmentacja jest zbędna a często szkodliwa.
- Czas dostępu to:
  - Czas wybrania urządzenia
  - Czas wybrania układu przez kontroler w urządzeniu,
  - Czas wprowadzenia adresu,
  - Czas przetwarzania bloków.
- Ze względu na fakt, że zapis i odczyt mogą odbywać się w blokach różnej długości, czasy te mogą być różne a nawet zależna od tego, czy wcześniej blok został skasowany czy nie (patrz opcja "TRIM").

Gospodarowanie SSD

- Pierwotnie ułożenie danych na dyskach solid-state realizowane było przez program obsługi systemu operacyjnego. Współcześnie zajmuje się tym kontroler dysku SSD o ile program obsługi zapewni mu odpowiednie informacje.

- Ogólnie, wybranie układu jest bardzo szybką operacją, szybszą niż adresowanie.
- Pliki więc są zapisywane (w **stronach**) po wszystkich układach przez kontroler. W ten sposób podczas odczytu:
  - Wszystkie układy są adresowane tym samym adresem równocześnie (czas jak adresowanie pojedynczego układu).
  - Wybierany i odczytywany jest pierwszy układ,
  - Później drugi, trzeci, itd.
- Dzięki temu mniej czasu pochłania adresowanie.

- Zapis jest bardziej kłopotliwy i w praktyce powolniejszy - nawet 10x wolniejszy niż odczyt! Stąd zapis często jest buforowany wielopoziomowo - najpierw przez system operacyjny, później w wewnętrznym buforze dysku SSD, opartym na specjalnej pamięci typu RAM. Wciąż możemy jednak go przyspieszać korzystając z adresowania wielu układów.

SSD: Zapis out-of-place

- Powiedzmy jednak, że chcemy tylko zmienić jakiś sektor w istniejącym zapisie...
  - Zapis strony nie może odbywać się bez wcześniejszego jej wyzerowania.
  - Zerowanie odbywa się „grupą" - tj. Po akurat zaadresowanym bloku z kilku stron.
  - Zapis odbywa się do bloku tj. zapis wszystkich układów pod danym adresem.
  - Blok może mieć kilka do kilkuset stron (**jeden chip fizyczny może mieć wiele układów logicznych**).
  - Preferuje się zapis do pustych bloków (wyzerowane maksimum stron), bo ich nie trzeba zerować.
- Czyli żeby zmienić ten ułamek strony, musimy:
  - Odczytać cały blok (zestaw stron ze wszystkich układów) do bufora,
  - Zmodyfikować odpowiednie miejsca w buforze,
  - Wyzerować blok,
  - Wgrać bufor do bloku ponownie.
- Co sprowadza wydajność poniżej tej z dysków magnetycznych.

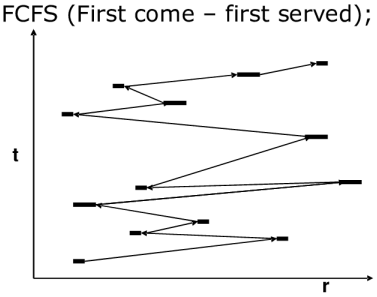
A może...

- **Oznaczyć** tylko poprzedni blok jako **nieważny** i zapisać nową wersję gdzieś indziej?
- Należy wówczas zapewnić jakiś mechanizm **translacji** numerów stron fizycznych/logicznych i zapisać taką tablicę na ssd.
  - PROBLEM: Dodatkowo pojawia się znaczna **fragmentacja** wewnętrzna bloków.
- "**Garbage collector**" - program będący częścią systemowego programu obsługi lub firmware dysku SSD, ograniczający nieważne strony i bloki - wyzwalany cyklicznie lub w przypadku zbyt dużej ilości użytych bloków względem niskiej zajętości bloków przez strony. GC zeruje bloki zawierające najmniej (lub 0) nieunieważnionych stron, może również przemieszczać strony - defragmentując na poziomie bloków i tworząc wolne bloki.
- **TRIM** - System operacyjny może przekazać GC informację, że bloki należące do usuniętego pliku są systemowi operacyjnemu niepotrzebne i GC może je zabrać do celów zerowania później.

- Więc uwaga co do trwałości:
  - Wartość "Total LBA written" w SMART jest znacznie niższa niż rzeczywista ilość zapisanych komórek pamięci, bo nie uwzględnia działania GC.
  - Stosunek zapisów "w układ scalony" do zapisów ze strony systemu operacyjnego nosi nazwę Write Amplification. Wysoki WA (wynikający z częstych modyfikacji i znacznego zapełnienia SSD) skraca żywotność dysku i zmniejsza wydajność (w każdej chwili może wejść GC).
  - Stąd SSD mają zawsze nieco miejsca zarezerwowane dla potrzeb GC (Overprovisioning).
  - A administrator może mu tego miejsca dodać nie wykorzystując w pełni SSD.

Planowanie I/O dla dysku

- FCFS (First come – first served);



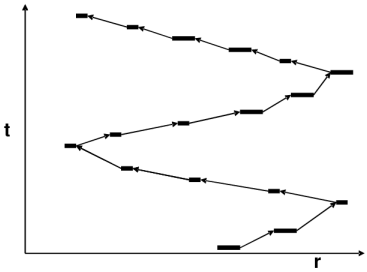
żądania wejścia-wyjścia realizowane w kolejności zgłaszania. Algorytm sprawiedliwy, ale mało wydajny. Przy dużej liczbie zgłoszeń planowanie zbliżone do losowego (najgorszego)

Planowanie I/O dla dysku

- PRI (priority) - krótkie zadania wsadowe i interakcyjne mają wyższy priorytet i pierwszeństwo w dostępie do dysku. Rozwiązanie nieefektywne dla baz danych.
- LIFO (last in-first out) - minimalizuje ruch głowicy (żądania korzystają z ograniczonego obszaru na dysku). Możliwość głodzenia procesów.
- SSTF (shortest seek time first) - najmniejszy ruch głowicy od pozycji bieżącej. Zawsze minimalny czas przeszukiwania. Możliwe głodzenie.

Planowanie I/O dla dysku

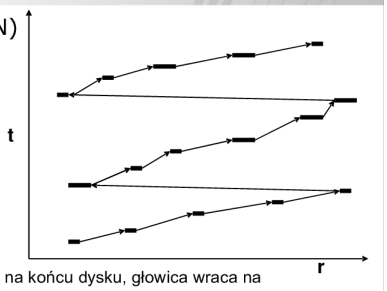
- SCAN



Głowica przesuwa się od jednej krawędzi dysku do drugiej, lub do ostatniego żądania. Potem zmienia się kierunek przesuwania. Działa jak winda - najpierw zamówienia „w górę", potem „w dół". Najszybciej po sobie obsługiwane żądania na skrajnych ścieżkach, a najpóźniej - po przeciwnej stronie dysku. Nie ma głodzenia.

Planowanie I/O dla dysku

- C-SCAN (circular SCAN)



Po obsłużeniu ostatniego zamówienia na końcu dysku, głowica wraca na początek i zaczyna od zera.
- N-step-SCAN - podział kolejki na podkolejki o długości N. Kolejne podkolejki obsługiwane metodą SCAN do końca.
- FSCAN - dwie kolejki, jedna realizowana, nowe żądania przychodzą do drugiej.

W porządku, ale dlaczego to wszystko często nie działa?

- Starsze dyski twarde mają swoją geometrię zapisaną w komputerze. Odpowiada ona dokładnie temu co znajduje się fizycznie w mechanice dysku (ilość talerzy, ścieżek na talerz, sektorów na ścieżkę). Np. C/H/S: 615/4/17 (cylindry/głowice/sektory na ścieżkę)
- System operacyjny korzysta z tych informacji i optymalizuje pracę z dyskiem.
- Ze względu na maksymalne wartości geometrii, nowoczesne dyski o dużej pojemności nie mogą być tak adresowane - po prostu dziesiątki tysięcy sektorów na ścieżkę **nie zmieści się w zmiennej**.
- Stosuje się **logiczne adresowanie** - elektronika dysku przelicza z innej geometrii, np. C/H/S: 16383/512/63 (256 talerzy - absurd).
- Tyle, że nie można tutaj łatwo przewidzieć ruchu głowic, więc omawiane algorytmy dadzą marginalny przyrost wydajności, a jak już to dla plików ciągłych.

System plików

- Na urządzeniach mamy różne systemy plików. Gdzie umieścić program tłumaczący je na abstrakcję plików systemu operacyjnego?
  - W jądrze - Windows, Unix, Linux (dawniej, po części) - limitowane możliwości korzystania z „obcych" systemów, trudniejsza implementacja nowych FS, ale za to szybciej działa na wolniejszych maszynach.
  - Przestrzeń użytkownika - Linux (coraz więcej), Mac OS (częściowo) - można użyć zewnętrznego programu do obsługi „obcych" systemów, łatwiej je zaimplementować, ale system musi oferować odpowiednie interfejsy by dało się mu „udostępnić" listy i operacje dla plików/katalogów.

Użytkowy interfejs I/O - drivery urządzeń

- Słowo STEROWNIK występuje w polskiej dokumentacji:
  - Jako „interface" sprzętowy (karta rozszerzeń, odpowiedni chip).
  - Jako „driver" programowy (program obsługi).
- Moduły sterujące - wewnętrznie dostosowane są do konkretnych urządzeń, a zewnętrznie udostępniają pewien standardowy interfejs.
- Standaryzacja pomaga producentom sprzętu tworzyć sterowniki do własnych urządzeń, widziane przez „obce" systemy operacyjne.
- Systemy muszą umożliwiać instalację sterowników do nowego sprzętu.

Zegary i czasomierze

- Spełniają trzy podstawowe funkcje:
  - Podawanie bieżącego czasu,
  - Podawanie upływającego czasu,
  - Powodowanie wykonania określonej operacji w określonej chwili.

Zegary i czasomierze

- **Programowalny RTC** - służy do pomiaru upływającego czasu i powodowania wykonania operacji w zadanym czasie,
  - Można go zaprogramować na określony czas, po którym generuje on przerwanie,
  - Jest to też zegar systemowy do taktowania kwantów czasu (dla przydziału procesora).
- **Programmable Interval Timer (PIT)** - służy do wykonywania okresowego zadań systemu, np. przełączanie kontekstu w starszych systemach.
  - Może wyzwalać przerwanie po jego zadziałaniu.
  - Działa na zasadzie licznika (przerwanie w momencie jego przeładowania) połączonego ze źródłem częstotliwości przez dzielnik (preskaler).

Zegary i czasomierze

- **HPET - High Precision Event Timer** - taktowany stabilnym sygnałem często szybszym niż PIT czy RTC.
  - Służy do wyzwalania przerwań bardzo często - np. przełączanie kontekstu współczesnych systemów.
  - Można użyć go do synchronizacji przetwarzanych multimediów.
- **Inne czasomierze** - Gdy w systemie nie ma łatwego dostępu do RTC, PIT a HPET zupełnie, systemy mogą używać czasomierzy systemu oszczędzania energii (okresowe sprawdzenie stanu zarządzania energią → przerwanie) lub licznika czasu w procesorze (Time Stamp Counter, TSC) co jest kłopotliwe - trudniej wyzwalać przerwania.

Urządzenia sieciowe

- Sieciowe wejście-wyjście różni się znacznie od dyskowego, pod względem wydajności i adresowania.
- Interfejs gniazda (socket) - aplikacje umożliwiają tworzenie gniazda, połączenie lokalnego gniazda ze zdalnym adresem, nasłuchiwanie, przesyłanie i odbieranie pakietów za pomocą połączenia.
- Funkcje systemowe zarządzają gniazdami (Windows - funkcja „wybór gniazda" z różnymi argumentami, Linux - zestaw funkcji)
- W systemach **Windows NT** - interfejs do kontaktowania się z kartą sieciową oraz interfejs do do protokołów sieciowych.
- W systemie **Unix** - półdupleksowe potoki, pełnodupleksowe kolejki FIFO, pełnodupleksowe strumienie, kolejki komunikatów i gniazda.

I/O w różnych systemach

- MS-DOS
  - Nazwy urządzeń zakończone dwukropkiem, np. A:, C:, PRN:
  - Odwzorowane są na określone adresy portów za pomocą tablicy urządzeń.
  - System operacyjny może przypisać każdemu urządzeniu dodatkowych funkcji, np spooling dla drukarki.
- UNIX
  - Nazwy urządzeń są pamiętane w przestrzeni nazw systemu plików (katalog /dev /devices).
  - W strukturze katalogowej zamiast numeru I-węzła jest zapisany numer urządzenia w postaci pary <starszy,młodszy>.
  - **Starszy** numer urządzenia identyfikuje moduł sterujący, który trzeba wywołać dla obsługi we-wy, a młodszy jest przekazywany do modułu sterującego jako indeks do tablicy urządzeń.
  - Wpis w tablicy urządzeń zawiera adres portu lub adres sterownika odwzorowany w pamięci.

I/O w Uniksie (c.d.)

- **Strumień** - kanał komunikacyjny pomiędzy procesem użytkownika a urządzeniem.
- Strumień składa się z:
  - głowy - interfejs z programem użytkownika,
  - zakończenia sterującego - interfejs z urządzeniem
  - modułów przetwarzających - filtrów na komunikatach płynących w strumieniu.
- Każdy ze składników strumienia zawiera co najmniej jedną parę kolejek: we i wy.
- Kontrola przepływu w strumieniu realizowana jest za pomocą buforów wejścia-wyjścia.
- We-wy na strumieniu jest asynchroniczne.
- Sterownik jest zawieszony do chwili pojawienia się danych.
- Gdy bufor jest pełny - utrata komunikatów (broken pipe).

Jądro Uniksa i I/O

- Urządzenia mogą być obsługiwane przez elementy jądra (Solaris, AIX) lub przez moduły (Linux) – ładowalne programy.
- Moduły nie muszą być cały czas załadowane – w pamięci mogą pozostać tylko te niezbędne dla danej konfiguracji sprzętowej.
- Moduły są hierarchiczne i od siebie zależne – np. aby załadować moduł obsługi czujników temperatury, wpierw ładowany jest moduł obsługi magistrali czujników (i2c).
- Użyteczne polecenia: lsmod, modprobe, rmmod.

Programy obsługi w Windows

- Podstawowe założenie: Komunikujemy się z urządzeniami przez ich interfejsy.
- np. Podłączamy pendrive do USB. Gdzie pendrive pojawi się w przestrzeni adresowej?
  - **Nie pojawi się**. Komunikacja odbywa się **przez magistralę USB.**
  - Czyli driver USB poda nowe urządzenie i system będzie mógł z niego korzystać używając drivera urządzenia wysyłającego komunikaty do drivera magistrali.
- Podobnie jest z wentylatorami/LEDami/przyciskami (magistrala SPI), kamerkami, drukarkami, skanerami (USB, protokoły sieciowe).
- Model sterowników w Windows odzwierciedla te zależności.

Programy obsługi w Windows

- Programy rezydentne – historycznie pierwsze, dziś nie stosowane.
- Moduły VxD – Programy są planowane przez system jak każda aplikacja, ale mają dostęp do pamięci na uprawnieniach jądra. Stanowią „pomost" między portami I/O w pamięci a wirtualnymi urządzeniami dla aplikacji systemu. Ze względów bezpieczeństwa dziś w zasadzie nie stosowane (z kilkoma wyjątkami).

WDM – Windows Driver Model

- Programy zorganizowane w stos, komunikują się ze sobą przesyłając wiadomości (IRP – I/O Request Packet).
- Urządzenie reprezentowane jest jako stos programów (inne urządzenie może wykorzystywać je w innej konfiguracji).
- Trzy rodzaje programów obsługi:
  - Bus driver – sterownik magistrali – odpowiada za bezpośrednie wysyłanie komunikatów z/do urządzenia i enumerację urządzeń.
  - Function driver – Oferuje aplikacjom możliwości odczytu/zapisu z/do urządzenia.
  - Filter driver – modyfikuje zachowanie urządzenia adaptując dane przesyłane przez aplikacje.
- Sterowniki są podpisane cyfrowo, choć sprawdzenie może zostać pominięte.

I/O w Windows

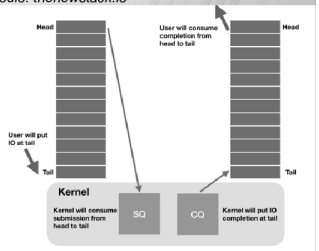
- **Cache manager** - zarządza całym podsystemem wejścia-wyjścia.
- Podsystem we-wy traktuje **File System Drivers** tak samo jak **Device Drivers**.
- **Network Drivers** - sterowniki sieci
- Hardware **Device Drivers** - dostęp do rejestrów urządzeń wejścia-wyjścia.
- Wejście-wyjście może być **asynchroniczne** i **synchroniczne**.
- System ma wbudowane mechanizmy do sygnalizowania zakończenia asynchronicznego wejścia-wyjścia.
- Możliwe uprawnienia dla każdego z urządzeń i portów I/O (domyślnie blokowane dla programów użytkownika – tylko przez sterowniki systemowe).

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, zakończonych 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.

io_uring … a może io_uring_spawn?

- Tworzenie nowych procesów lub wątków taką samą metodą jak io_uring tworzy operacje I/O.
- Czasami niezbędne jest uruchomienie wątku lub procesu powiązanego z daną operacją, np. na serwerze rozpoczyna pracę sesja nowego użytkownika.
- Czy czekać na pełny fork() serwera?
- Zaproponowane 09.2022.
- Do implementacji po wersji 6.0

Dziękuję za uwagę