AGH

Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Systemy operacyjne
Wykład 04N

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 plików

AGH

System plików

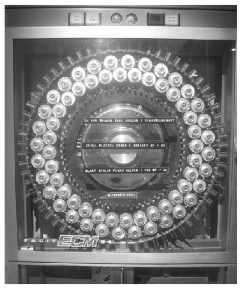
- **Plik** jest logiczną jednostką magazynowania informacji w pamięci nieulotnej.
- **Plik** jest nazwanym zbiorem powiązanych ze sobą informacji zapisanym w pamięci pomocniczej.
- **Plik** jest ciągiem bitów, bajtów, wierszy lub rekordów.

- Atrybuty pliku:
  - Nazwa (zgodna z regułami dla danego systemu operacyjnego),
  - Typ (jeżeli system tego wymaga),
  - Położenie (wskaźnik do urządzenia i położenie na tym urządzeniu),
  - Rozmiar (w bajtach, słowach lub blokach),
  - Ochrona (prawa dostępu),
  - Czas, data, identyfikator właściciela,
  - Ewentualnie inne metadane (fork zasobów (Apple), rekord metadanych (BeFS)).

AGH

Rys historyczny

- Komputery bez dysków, oparte o taśmy: Urządzenie jest plikiem.
- Później: Na urządzeniu może być kilka plików, ale i tak chodzi o zmianę taśm. Ręczną lub automatyczną (projekt Facit Carousel),
- Później: Na jednej taśmie może być kilka plików. Konieczność zwiększenia trwałości nośników taśmowych (próżnia i.t.p. rozwiązania),
- Później: Taśmy indeksowane.



- Systemy dyskowe: Dysk posiada prostą listę plików.
- Później: systemy plików.

AGH

Rys historyczny

- Lata 1970-80 - mikrokomputery:
  - Urządzenie (np. Magnetofon, drukarka, napęd - jest plikiem)
  - Później: Na urządzeniu o dostępie swobodnym może być pewna ilość osobnych plików. Poprzedni punkt pozostaje w mocy odnośnie napędów taśmowych i niektórych urządzeń I/O.
  - W takim przypadku systemem plików steruje oprogramowanie układowe napędu dyskowego - mało wydajne, ale jedyne możliwe bez ładowania dodatkowego sterownika z taśmy.
  - Maszyny z systemem CP/M: Stacja dysków staje się komputerem. Mikrokomputer staje się terminalem.

AGH

Rys historyczny

- Uniksowe mainframe: Hierarchiczny system plików.
- IBM PC (DOS 1.x) - prosta lista plików na dyskietce.
- IBM PC (DOS 2.x) - katalogi. Dyski twarde do 20-30MB (lub więcej ze sterownikami), kwestia systemu plików na większych dyskach nie jest ustandaryzowana.
- DOS 3.x-6.x - FAT - pełna obsługa zagnieżdżonych katalogów (z pewnymi limitami) i wielu partycji.
- Windows 95 - VFAT i 95 OSR2 OSR2 - FAT32 - większe dyski, rozszerzenie FATa tak, by plik miał nazwę dłuższą niż 8+3.
- Windows NT - NTFS - w systemie plików pojawiają się strumienie, rozszerzone atrybuty, sparse files, obiekty (OFS - nie dostał się do końcowej wersji WinNT).

Operacje plikowe

- **Tworzenie pliku**:
  - znalezienie miejsca w systemie plików
  - wpis do katalogu
- **Zapisywanie pliku** - podaje się nazwę (identyfikator) pliku i informację do zapisania. Istotne jest miejsce od którego piszemy (wskaźnik położenia).
- **Czytanie pliku** - podaje się nazwę pliku i bufor w pamięci. Można wykorzystać ten sam wskaźnik położenia.
- **Zmiana pozycji w pliku** - modyfikacja wskaźnika położenia.
- **Usuwanie pliku** - zwalnia się przestrzeń zajmowaną przez plik i likwiduje się wpis katalogowy.
- **Skracanie pliku** - likwidowanie części albo całej zawartości pliku bez kasowania jego nazwy i atrybutów.

Inne operacje

- **Dopisywanie** - dopisywanie nowych informacji na końcu istniejącego pliku.
- **Przemianowanie pliku** - zmiana nazwy pliku, często tą samą komendą wykonuje się przesuwanie pliku, czyli zmianę jego położenia - do innego katalogu, na inny dysk.
- **Otwieranie pliku** - stosowane w wielu systemach w celu uniknięcia wielokrotnego czytania informacji o pliku - dane z katalogu kopiowane są do tablicy otwartych plików.
- **Zamykanie pliku** - kiedy plik przestaje być potrzebny, usuwa się wpis z tablicy otwartych plików.

- **Otwieranie i zamykanie plików w systemach wieloużytkownikowych musi uwzględniać równoczesne korzystanie z pliku przez kilka procesów!**

Typy plików

- System rozpoznaje typy plików poprzez:
  - Rozszerzenia - w MSDOS niektóre typy plików określane przez rozszerzenia nazwy (*.com, *.exe…),
  - Liczby magiczne - oraz typowe fragmenty początku pliku - identyfikacja w systemie Unix (komenda file, plik /etc/magic),
  - Atrybut twórcy (w MAC OS) - czyli nazwę programu, przy pomocy którego utworzono plik.

Typy dostępu do plików

- **Dostęp sekwencyjny** - informacje w pliku są przetwarzane kolejno, rekord po rekordzie,
- **Dostęp bezpośredni** - umożliwia czytanie z zapisywanie bloków w dowolnej kolejności. Rekordy muszą być stałej długości. Używany jest tam, gdzie potrzebny jest szybki dostęp do wielkich ilości informacji, np w bazach danych.
- **Dostęp indeksowy** (plik indeksowy w pamięci, lub na dysku)

Katalogi

- **Jednopoziomowy** - ograniczeniem jest konieczność spełnienia warunku niepowtarzalności nazw.
- **Dwupoziomowy** - każdy użytkownik ma własny katalog macierzysty, a w nim pliki.
- **Wielopoziomowe drzewiaste**.
- **Acykliczne grafy** - do pliku można dojść wieloma drogami.

Ochrona plików

- Można kontrolować wiele operacji:
  - **czytanie** pliku
  - **pisanie** do pliku, lub zapisywanie go na nowo
  - **wykonywanie** - załadowanie pliku do pamięci i wykonanie go
  - **dopisywanie** danych na końcu pliku
  - **usuwanie** pliku i zwalnianie obszaru przez niego zajętego
  - **opisywanie** - wyprowadzenie nazwy i atrybutów pliku
- Klasy użytkowników pliku:
  - **właściciel** - użytkownik, który utworzył dany plik
  - **grupa** użytkowników, którzy wspólnie korzystają z pliku i potrzebują podobnego zakresu dostępu
  - **wszyscy** inni.

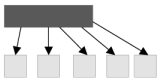
Przydział miejsca na dysku

- **Przydział ciągły** - każdy plik zajmuje ciąg kolejnych bloków na dysku.
  - zalety: minimalna liczba operacji przeszukiwania dysku, łatwość implementacji dostępu sekwencyjnego i bezpośredniego.
  - wady: trudności ze znalezieniem wolnego miejsca (fragmentacja zewnętrzna),

- **przydział listowy** - istnieje lista powiązanych ze sobą bloków dyskowych, stanowiących dany plik. Bloki te mogą się znajdować w dowolnym miejscu na dysku.
  - zalety:brak fragmentacji zewnętrznej, nie trzeba deklarować długości pliku (plik może rosnąć, dopóki są wolne bloki)
  - wady: trudność w implementacji dostępu bezpośredniego, zajęcie sporej przestrzeni przez wskaźniki, w przypadku błędu jednego wskaźnika można wejść w obszar innego pliku.

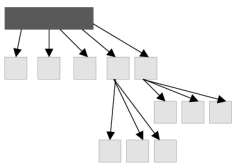
Przydział miejsca na dysku

- **Przydział indeksowy** - podobny jak przydział listowy, ale wskaźniki umieszczone w jednym miejscu - w tablicy indeksów
  - zalety: jak w przydziale listowym,
  - wady: wskaźniki bloku indeksowego zajmują zazwyczaj więcej miejsca niż wskaźniki przy przydziale listowym



Dla dużych plików...

- Schemat listowy - jeśli lista bloków jest dłuższa niż blok indeksowy, na ostatniej pozycji w bloku indeksowym podaje się adres bloku kontynuacji,
- Indeks wielopoziomowy - pozycje bloku indeksowego wskazują na bloki indeksowe poziomu drugiego.
- Schemat mieszany -pierwsze kilka, kilkanaście pozycji wskazuje bezpośrednio na bloki, a następne 2-3 na indeksy poziomu drugiego w indeksowaniu 2,3,4-poziomowym. Schemat ten zastosowany w systemie UNIX.



Przydział miejsca a wydajność

- Przykład złego doboru metody przydziału: przy przydziale listowym i dostępie bezpośrednim, dostęp do bloku n wymaga n dostępów do dysku.
- Metody poprawy wydajności
  - deklaracja typu dostępu przy tworzeniu pliku - jeśli dostęp ma być bezpośredni, stosuje się przydział ciągły (wymaga podania wielkości pliku przy tworzeniu)

    Jeśli dostęp ma być sekwencyjny, stosuje się przydział listowy

    System musi mieć zaimplementowane obie metody przydziału
  - Typ przydziału zależny od wielkości pliku - dla małych, kilku-blokowych plików przydział ciągły, dla dużych plików przydział np indeksowy.
  - Stosowanie klastrów (gron) i różnych wielkościach i stosowanie możliwie największych (np 64 kB) dla dużych plików. Uzupełnianie do końca pliku małymi klastrami.

Zarządzanie wolną przestrzenią

- **mapa bitowa** - w wektorze bitowym każdy wolny blok jest reprezentowany przez 1 a zajęty - przez 0. łatwość znalezienia wolnego bloku istnieje dzięki rozkazom procesora pokazującym pozycję pierwszego niezerowego bitu w słowie. Metoda ta może być stosowana dla małych dysków.
- **lista powiązana** - w pamięci przechowuje się położenie pierwszego wolnego bloku, a w nim - położenie następnego itd. Metoda mało wydajna - aby przejrzeć listę wolnych bloków, należy wszystkie przeczytać.
- **grupowanie** - w pierwszym wolnym bloku przechowywana jest lista n wolnych bloków. W n-tym wolnym bloku znajduje się lista następnych n bloków itd.
- **zliczanie** - przechowuje się adres pierwszego wolnego bloku i liczbę n następujących po nim wolnych bloków. I tak dla każdej grupy. N jest zazwyczaj > 1

Poprawa wydajności systemów dyskowych

- **pamięć podręczna** - przechowywanie całych ścieżek dysku w pamięci - prawdopodobnie będą z nich w niedługim czasie czytane dane. Wykorzystana do tego celu specjalna pamięć, lub nieużywana pamięć główna.
- **wczesne zwalnianie** - usuwanie bloku z bufora natychmiast, gdy pojawia się zamówienia na następny (oszczędza pamięć)
- **czytanie z wyprzedzeniem** - z zamówionym blokiem czyta się kilka następnych, gdyż prawdopodobnie zaraz będą potrzebne.
- **RAM-dysk** - wszystkie operacje dyskowe przeprowadza się w pamięci. Zawartość RAM-dysku jest pod kontrolą użytkownika. Wada - zawartość ginie po wyłączeniu zasilania, awarii.
- **Opóźniony zapis** – Zapis z pamięci podręcznej następuje później, preferuje się nawet zapis na dysk dopiero wtedy gdy potrzeba te dane odczytać. Dzięki temu opóźnia się zbędne zapisy danych tymczasowych

Czynności operatorskie

- **Sprawdzanie spójności** - po awarii systemu, czy np po wyłączeniu „z kontaktu”. Program chkdsk (Windows), scandisk (DOS), fsck (UNIX). Zazwyczaj uruchamiają się automatycznie (znacznik „czystości” systemu plików)

- **Składowanie i odtwarzanie** - robienie kopii systemu plików na innym nośniku i odtwarzanie po awarii. Kopia zapasowa, Norton Ghost (DOS, Windows), tar, backup, restore (Unix).
- **Składowanie przyrostowe** - kopia całego systemu raz, a potem zapisywanie tylko zmienionych plików.
  Kopie „wieczyste” - co jakiś czas, taśmy pozostają w archiwum „na zawsze”.

System Plików - DOS FAT

- **Katalog** - zawiera: nazwy plików (8 znaków), rozszerzenie (3 znaki), długość pliku, atrybuty (h s r a), datę utworzenia pliku, wskazanie pozycji FAT
- **FAT** (file allocation table) - tablica, której elementy odpowiadają kolejnym jednostkom alokacji (sektorom, blokom, klastrom). FAT jest umieszczony na początku dysku, w dwóch kopiach.

Przykład użycia: Plik Z1 zajmuje bloki: 7,8,11,3, Z2 zajmuje blok 4, a Z3 jest w blokach: 1,2,5,6,9

```
Nr:  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 15 17 18 19 20
    02 05 ff ff 06 09 08 11 ff 00 03 00 00 00 00 00 00 00 00 00
```

FAT 12 - każda pozycja miała 12 bitów, czyli mogła zawierać wartość 0-4096, co przy klasterach 8 kB dawało 32 MB

Kolejne wersje systemu - FAT 16 (do 2 GB) i FAT 32.
Bariera - czas przeszukiwania tablicy FAT.

System Plików – NTFS (Windows)

- **Wolumin** (volume)- podstawowa jednostka dyskowa - może to być część dysku, cały dysk lub kilka dysków razem
- **Klaster** (grono, cluster) - podstawowa jednostka przydziału, jest to grupa sąsiadujących sektorów. Są znacznie mniejsze niż w systemach z FAT - 4 kB dla dużych dysków. Jako adresy dyskowe używane są logiczne numery klastrów.
- **Plik** jest obiektem strukturalnym złożonym z atrybutów. Atrybuty pliku są strumieniami bitów. Jednym z atrybutów są też dane pliku.
- **Główna tablica plików** (MFT) - przechowuje opisy plików, zawarte w jednym lub kilku rekordach dla każdego pliku.
- **Odsyłacz do pliku** - niepowtarzalny identyfikator pliku, składa się z 48-bitowego numeru pliku (pozycja w MFT) i 12- bitowego numeru kolejnego.

NTFS – inne pliki systemowe

- **Kopia MFT** - kopia pierwszych 16 pozycji MFT - do działań naprawczych,
- **Plik dziennika** - zawiera wszystkie zmiany danych w systemie plików,
- **Plik woluminu** - dane woluminu, dane o wersji NTFS, który go sformatował, bit informujący o konieczności chkdsk,
- **Tablica definicji atrybutów** - typy atrybutów i dopuszczalne dla nich operacje,
- **Katalog główny**
- **Plik mapy bitów** - wskazuje zajęte i wolne klastry,
- **Plik rozruchowy** - kod początkowy NT,
- **Plik uszkodzonych klastrów**,

- **Usuwanie skutków awarii** jest stosunkowo proste, gdyż operacje dyskowe odbywają się na zasadzie **transakcji**.

NTFS – nazwy plików systemowych

System plików	Nazwa Pliku	Rekord MFT	Funkcje
Master file table	$Mft	0	Zawiera podstawowe informacje na temat plików i folderów w systemie plików.
Master file table 2	$MftMirr	1	Kopia 4 pierwszych rekordów MFT.
Log file	$LogFile	2	Zapamiętuje ostatnie transakcje wykonane na systemie plików.
Volume	$Volume	3	Zawiera informacje nt. woluminu: nazwa, wersja.
Attribute definitions	$AttrDef	4	Tablica atrybutów, ich nazw oraz opisu.
Root file name index	$	5	Katalog główny.

NTFS – Nazwy plików systemowych

System plików	Nazwa Pliku	Rekord MFT	Funkcje
Cluster bitmap	$Bitmap	6	Przechowuje informacje o używanych klastrach.
Boot sector	$Boot	7	Boot sector
Bad cluster file	$BadClus	8	Zawiera informacje o niedziałających klastrach.
Security file	$Secure	9	Deskryptory praw dostępu.
Upcase table	$Upcase	10	Zamienia małe litery na wielkie, zgodnie z Unicode.
NTFS extension file	$Extend	11	Używana w wielu fuknjach NTFS, np.: quota, reparse point, object ID.

NTFS – system plików Windows

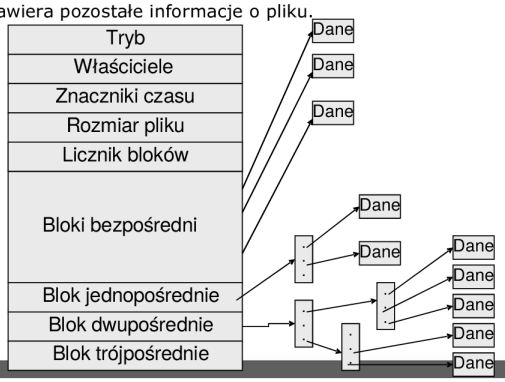
- Każda operacja jest wykonywana na zasadzie transakcji.
- Najpierw zmiany dokonywane na metadanych zostają zapisane w dzienniku, dopiero potem faktyczna operacja ma miejsce i jest zatwierdzana.
- W przypadku awarii systemu, uruchamiane jest odzyskiwanie systemu plików.
- Przeglądany jest dziennik i wszystkie niedokończone transakcje są usuwane bądź realizowane zgodnie z zapamiętanymi tam informacjami.
- Takie rozwiązanie gwarantuje stabilność systemu plików, jednak czasami można utracić część informacji, które były modyfikowane w trakcie awarii, bądź znajdowały się na zepsutym klastrze.

NTFS

- W przypadku gdy system plików odkryje niedziałające klastry na dysku, automatycznie oznacza je jako zepsute i podstawia na jego miejsce nieużywany klaster.
- W przypadku odwołania do adresu podmienionego klastra, NTFS realizuje żądanie na nowym zmapowanym klastrze.
- W przypadku odczytu z błędnego klastra traci się informacje na nim zawarta, jednak w przypadku zapisu, użytkownik nie dowie się nawet, że wystąpił po drodze jakiś błąd.
- Powtarzające się awarie klastrów są wskazówką do wymiany nośnika na nowy.

Systemy plików UNIX

- Katalog - zawiera: nazwy plików (w starszych do 256 znaków), wskazanie Inode (węzła). Za wyjątkiem znacznika, nie różni się od pliku.
- I-node - zawiera pozostałe informacje o pliku.



Zawartość I-węzła

Typ	Pole	Opis
__u16	i_mode	typ pliku (dowiązanie symboliczne, zwykły plik, katalog, urządzenie znakowe, urządzenie blokowe, gniazdo, kolejka FIFO) i prawa dostępu
__u16	i_uid	Identyfikator właściciela pliku
__u32	i_size	Długość pliku w bajtach
__u32	i_atime	Czas ostatniego dostępu (w sekundach od epoki Uniksa)
__u32	i_ctime	Czas ostatniej zmiany i-węzła (jw.)
__u32	i_mtime	Czas ostatniej zmiany zawartości pliku (jw.)
__u32	i_dtime	Czas usunięcia pliku (jw.)

Zawartość I-węzła (2)

Typ	Pole	Opis
__u16	i_gid	Identyfikator grupy
__u16	i_links_count	Licznik twardych dowiązań do pliku
__u32	i_blocks	Liczba bloków danych pliku (po 512 bajtów)
_u32	i_flags	Flagi pliku ("tylko dodawanie (append only)", "nie można zmieniać (immmutable)", i inne)
union	osd1	Specyficzne informacje systemu operacyjnego
__u32	i_block	Wskaźniki do bloków danych (zwykle 15, pierwszych 12 to wskaźniki bezpośrednie, jeden pośredni, jeden podwójnie pośredni, jeden potrójnie pośredni)

Zawartosc I-węzła (3)

Typ	Pole	Opis
__u32	i_version	Wersja pliku (dla NFS)
__u32	i_file_acl	Lista kontroli dostępu do pliku (ACL)
__u32	i_dir_acl	Lista kontroli dostępu katalogu
__u32	i_faddr	Adres fragmentu
union	osd2	Specyficzne informacje systemu operacyjnego

System plików UNIX

- **Superblok** - zawiera statyczne parametry systemu plików: całkowity rozmiar systemu plików, rozmiary pełnych bloków danych i ich fragmentów, dane dotyczące przydziału miejsca.
- **Grupa cylindrów** - dla zminimalizowania ruchu głowic dysku, cylindry dysku (ta sama ścieżka na wszystkich głowicach) pogrupowano po kilka, zapisując w każdej grupie:
  - superblok,
  - blok cylindra (zawierający mapę wolnych bloków, mapę wolnych i-węzłów, dane statystyczne do strategii przydziału),
  - I-węzły,
  - bloki danych (do końca grupy cylindrów).

System plików Ext4

- Domyślny system plików większości dystrybucji systemu Linux.
- Następca Ext3, Ext2, Ext, Minix FS…
- Możliwości:
  - Urządzenia do 1EB (Eksabajt),
  - Nielimitowana głębokość ścieżek katalogów,
  - Dziennik transakcji z obsługą sum kontrolnych,
  - API szyfrowania w trakcie operacji dyskowych,
  - Opóźnienie czyszczenia i-nodeów do momentu gdy zachodzi potrzeba ich zapisu.
  - Opóźnienie problemu roku 2038 o dwa bity (do roku 2446).
  - Przydziały miejsca na poziomie systemu plików, bardziej elastyczne niż partycjonowanie.

Ext4 - przydział miejsca

- Przestrzeń przydzielana jest nie w blokach, lecz w strukturach definiujących zakresy (extents).
- W metadanych jednego pliku jest miejsce na 4 extenty, maksymalny rozmiar 128MB * 4 = 512MB
- Większy plik - extenty są przechowywane w drzewiastej strukturze umożliwiającej wydajne poszukiwanie po offsecie. (najczęściej do pliku dostęp dokonujemy albo odczytując od początku do końca, albo od jakiegoś przesunięcia)

Ext4 - walka z fragmentacją

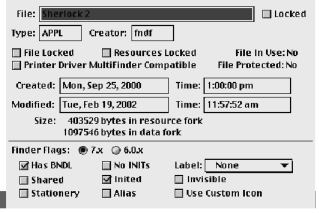
- Allocate-on-flush - Alokacja bloków dopiero w momencie zapisu bufora na dysku - zmniejsza fragmentację.
- Prealokacja - Możliwe jest jednokrotne alokowanie bloków dla pliku przed jego zapisem - dzięki temu dwa jednoczesne zapisy nie powodują problemów z fragmentacją gdy usunięto jeden plik.
- Nowe pliki zapisywane są tak, by dane były równomiernie rozmieszczone.
- Blokowanie niewielkiego % przestrzeni dyskowej celem takiego zapisywania (lub przepisywania) danych, by zminimalizować fragmentację (często 1-5%).

Ext4 - katalogi

- Specjalizowana forma drzewiastej struktury H-tree
- Istotną cechą w gałęzi B-drzewa jest nie wartość, a **zakres** wartości, jakiego możemy spodziewać się poniżej.
- Dochodzimy tak do najniższych elementów zawierających konkretne wartości.
- Dzięki temu możliwe jest bardzo szybkie zachowywanie i przetwarzanie identyfikatorów katalogów.

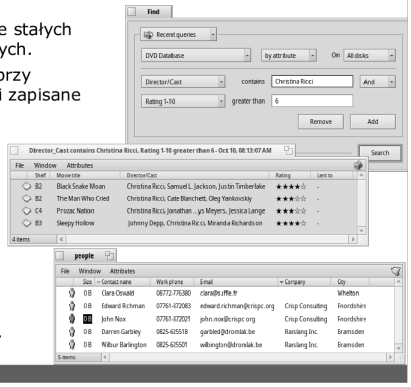
Więcej metadanych! Apple HFS i BeFS

- Apple HFS i późniejsze: Każdy plik lub katalog posiada "resource fork" – strukturę danych opisujących jak plik ma być prezentowany i traktowany przez system:
  - Czy obiekt jest łączem do innego obiektu,
  - Ikona i jej podstawowy kolor,
  - Program jaki otwiera dany plik ("Creator"),
  - Położenie ikony w oknie,
  - Czy obiekt nie posiada wyświetlanej nazwy,
  - Komentarz tekstowy,
  - Rozmiar/położenie okna programu otwierającego plik,
  - I wiele innych, w tym sporo nieużywanych.
- W Mac OS X powoli rezygnowano z forków kosztem ukrytych, niewyświetlanych plików z opisami.



BeFS

- Zaprojektowany w 1996 roku dla systemu BeOS, używany w Haiku OS.
- Wykorzystano extenty jak przy systemach Ext, składające się ze stałych ilości fizycznych bloków dyskowych.
- Są one jednak „pakowane” jak przy systemach uniksowych - w bloki zapisane bezpośrednio, jednopośrednio, dwupośrednio itd.
- Każdy plik może posiadać dowolną ilość i dowolny typ metadanych. Są one zapisane w dodatkowych extentach opisanych w rekordzie pliku.
- Strukturę metadanych dobiera się po typie pliku.
- Dzięki temu systemu plików można używać jak bazy danych.



MS WinFS, NewFS, OFS, ???

- Nigdy nie pojawił się w ostatecznej wersji Windows.
- Relacyjna baza danych jako system plików,
- Plik: Dane, metadane, zasoby i sposób otwarcia.
  - Sposób otwarcia: Otwierający program i systemowy moduł konwersji.
- Dzięki temu uwalniamy się od pułapki niezgodności formatów...
- ...stąd skupiamy się na danych i relacjach między nimi.
- Program jest w stanie "na żywo" generować strukturę plikową (jak np.. Uniksowy ProcFS) oferując w ten sposób "chmurowe" dyski, E-mail, dokumenty, inne komputery.

Windows 95: Klient poczty Exchange udaje folder

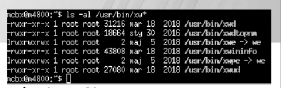
- Problemy:
  - Ciągła ewolucja formatów plików i duże różnice między ich możliwościami.
  - Niezgodności międzyplatformowe.
  - Konieczność zrezygnowania ze zgodności wstecznej.

Project Longhorn (Vista): Folder jako kalendarz →

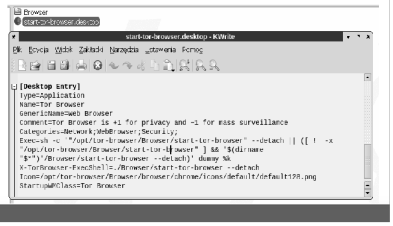
Skróty

- Większość współczesnych systemów plików organizuje pliki w wielopoziomowy, hierarchiczny sposób (drzewo katalogów). W tych systemach jednak mogą istnieć wyjątki.
- W MS Windows możliwe jest „podmontowanie” katalogu do innego katalogu - linki. Bardziej popularną formą są realizowane na poziomie graficznego interfejsu skróty, czyli pliki o rozszerzeniu .lnk interpretowane odpowiednio przez graficzną powłokę.
- Siłowe zrealizowanie „skrótu do skrótu” spowoduje (we wcześniejszych wersjach powłoki Windows) zrestartowanie powłoki lub (w późniejszych wersjach) jedynie sprawdzenie obecności docelowego elementu, ewentualnie (w najnowszych wersjach) bezwzględne zwrócenie błędu bez dostępu do ryzykownego pliku na prawach systemu.

Skróty w Unix / Linux



- Soft link - łącze „miękkie” - uznawane przez konsolę i powłoki graficzne, pod pełną ich kontrolą. Jest osobnym obiektem (plikiem/folderem) posiadającym swój identyfikator. Kasując docelowy obiekt uszkadzamy link i tracimy dane. Tworzymy przez **ln -s źródło cel**
- Hard link - stanowi jedynie kolejne odniesienie do istniejącego identyfikatora **pliku**. Kasując jeden hard link wciąż możemy dostać się do pliku innym hard linkiem - dane tracimy dopiero po usunięciu ostatniego. Powłoka nie ma bezpośredniej możliwości kontroli hard linków. Tworzymy przez **ln źródło cel**
- Plik .desktop - stanowi pełny zapis wyglądu i zachowania docelowego obiektu. Nazwa, objaśnienie w menu powłoki, ikona, program docelowy i jego parametry, a także otwierane typy plików i ewentualnie tłumaczenia nazwy i opisu. Działa w związku z tym **wyłącznie** w środowisku graficznym. Tworzenie - jak pliku tekstowego.



System wejścia-wyjścia

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.
- Program użytkownika korzysta z wywołań BIOSu lub procedur w pamięci ROM urządzenia, co może być mniej lub bardziej standardowe - jw.
- 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.

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.

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).

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 (dysk, płyta)
- 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 sterownika:

Sterownik posiada rejestry do pamiętania danych i sygnałów sterujących. Procesor pisze i czyta do rejestrów w sterowniku.

- 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 (nóżkę) 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 (w starych konstrukcjach - przeplot!),
- **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

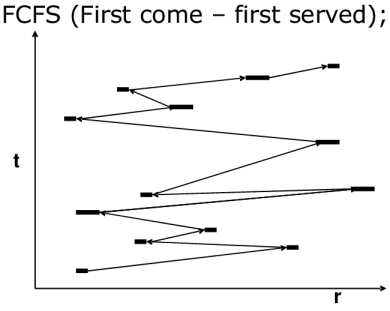
- Powiedzmy jednak, że chcemy tylko zmienić jakiś sektor w istniejącym zapisie...
  - Zapis strony nie może odbywać się bez wcześniejszego jej wyzerowania.
  - Zapis odbywa się do bloku tj. zapis wszystkich układów pod danym adresem.
  - Blok może mieć kilka do kilkuset stron (jeden chip fizyczny może 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);



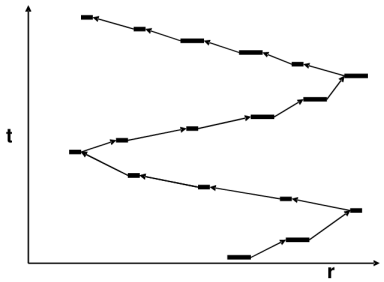
żądania wejścia-wyjścia realizowane w kolejności zgłaszania. Algorytm sprawiedliwy, ale mało wydajny. Przy dużej liczbie zgłoszeń planowanie zbliżone do losowego (najgorszego)

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

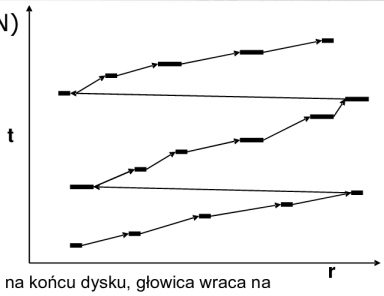
- SCAN



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.

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.
- **Czasomierz programowalny** - 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).

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.
- Funkcja **wybierz** zarządza gniazdami,
- 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

- 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.

Stos sieciowy w Windows

- Dzięki filtrom, programy obsługi działają dla **klientów, kart, protokołów i usług.**
  - **Karta** – karta sieciowa, rzeczywiste lub wirtualne urządzenie.
  - **Protokół** – sposób komunikacji przez sieć.
  - **Usługa** – realizacja zadania korzystając z protokołu.
  - **Klient** – aplikacja sieciowa udostępniająca możliwości sieci systemowi, możliwe jest tu wykorzystanie zewnętrznych programów.
- System odpowiada za prawidłowe połączenie tych elementów.

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).

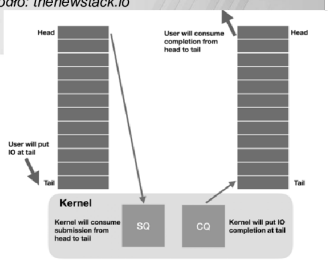
I/O Uring w Linuksie

- 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