KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Systemy operacyjne

Marek Grochowski
Jacek Kobus

Wydział Fizyki, Astronomii i Informatyki Stosowanej UMK (2022/2023)

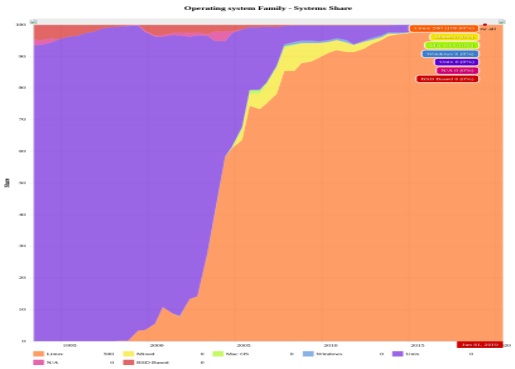
`https://jkob.fizyka.umk.pl/students/notes/notes.html (so[4].pdf)`
`git clone ssh://<user>@ameryk.fizyka.umk.pl/git/tm`

ostatnia aktualizacja: 8-12-2022

TOP500: 5 najszybszych superkomputerów (6/2021)

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, **Fujitsu** RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM** DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM / NVIDIA / Mellanox** DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, **NRCPC** National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, **HPE** DOE/SC/LBNL/NERSC United States	706,304	64,590.0	89,794.5	2,528

TOP 500: udział systemów operacyjnych (6-2020)[1]



[1] https://www.top500.org/statistics/overtime/

Windows kontra Linux wg E.S.Raymonda[2]

ESR na swoim blogu zatytułowanym *Last phase of the desktop wars?* napisał m.in:

- The two most intriguing developments in the recent evolution of the Microsoft Windows operating system are Windows System for Linux (WSL) and the porting of their Microsoft Edge browser to Ubuntu.

  [...] WSL allows unmodified Linux binaries to run under Windows 10. No emulation, no shim layer, they just load and go.

- Microsoft developers are now landing features in the Linux kernel to improve WSL. [...] To understand why, we need to notice how Microsoft's revenue stream has changed since the launch of its cloud service in 2010.

- Ten years later, Azure makes Microsoft most of its money. The Windows monopoly has become a sideshow, with sales of conventional desktop PCs (the only market it dominates) declining.

- Proton is the emulation layer that allows Windows games distributed on Steam to run over Linux.

... the end state this all points at is: New Windows is mostly a Linux kernel, there's an old-Windows emulation over it, but Edge and the rest of the Windows user-land utilities don't use the emulation. The emulation layer is there for games and other legacy third-party software.

[2] 25/09/2020: https://esr.ibiblio.org/?p=8764

Program wykładu

1. Wprowadzenie
   - Co to jest system komputerowy?
   - Co to jest system operacyjny?
   - Historia komputerów i systemów operacyjnych
   - Architektura współczesnych komputerów
2. Jak system komputerowy wykonuje programy?
   - Monitor prosty
   - Buforowanie
   - Spooling
   - Wieloprogramowość
   - Systemy z podziałem czasu
   - Systemy rozproszone, systemy czasu rzeczywistego

3. Struktura systemu komputerowego
   - Systemy z obsługą przerwań
   - Struktura wejścia-wyjścia
   - Dualny tryb pracy
4. Procesy
   - Model procesu i jego realizacja
   - Proces z poziomu powłoki
   - Zarządzanie procesami
   - Komunikacja międzyprocesowa
   - Synchronizowanie procesów. Zakleszczenia.

5. Zarządzanie pamięcią
   - Zarządzanie pamięcią bez wymiany i stronicowania
   - Wymiana
   - Pamięć wirtualna
   - Segmentacja
6. Zarządzanie przestrzenią dyskową
   - Rodzaje plików
   - Partycje i systemy plików, systemy plików z kroniką
   - Zarządzanie logicznymi wolumenami
   - Macierze dyskowe

7. Struktura systemów operacyjnych
   - Systemy monolityczne
   - Systemy warstwowe
   - Maszyny wirtualne
   - Model klient-serwer
8. Przykłady systemów operacyjnych
   - MSDOS, Windows 95/98
   - Windows 2000/NT/...
   - Unix, GNU/Linux

Literatura

[1] M. J. Bach. *Budowa systemu operacyjnego UNIX®*. Wydawnictwo Naukowo-Techniczne, Warszawa, 1995.

[2] D. P. Bovet i M. Cesati. *Linux Kernel*. Wydawnictwo RM, Warszawa, 2001.

[3] J. Glenn Brookshear and Denis Brylow. *Informatyka w ogólnym zarysie*. Wydawnictwo Naukowe PWN, Warszawa, 2022.

[4] R. Love. *Linux kernel. Przewodnik programisty*. Wydawnictwo Helion, Gliwice, 2004.

[5] L. Null i J. Lobur. *Struktura organizacyjna i architektura systemów komputerowych*. Wydawnictwo Helion, Gliwice, 2004.

[6] A. Silberschatz i P. B. Galvin. *Podstawy systemów operacyjnych*. Wydawnictwo Naukowo-Techniczne, wyd.5, Warszawa, 2002.

[7] W. Stallings. *Organizacja i architektura systemu komputerowego (wydanie XI)*. Wydawnictwo Naukowe PWN, Warszawa, 2022.

[8] W. Stallings. *Systemy operacyjne*. Wydawnictwo Robomatic, Wrocław, 2004.

[9] K. Stencel. *Systemy operacyjne*. Wydawnictwo PJWSTK, Warszawa, 2004.

[10] A. S. Tanenbaum and H. Bos. *Systemy operacyjne*. Wydawnictwo Helion, Gliwice, 2010.

[11] A. S. Tanenbaum. *Modern Operating Systems*. Prentice-Hall International, Upper Saddle River, 1992.

[12] E. S. Raymond. *UNIX Sztuka programowania*. Wydawnictwo Helion, Gliwice, 2004.

[13] M. Bar. *Linux – systemy plików*. Wydawnictwo RM, Warszawa, 2002.

[14] *Free OnLine Books*. http://www.linux.org/docs/online_books.html/.

[15] *Interactive Linux Kernel Map*. http://www.linuxdriver.co.il/kernel_map/.

[16] R. Ligonnière. *Prehistoria i historia komputerów*. Zakład Narodowy im. Ossolińskich – Wydawnictwo, Wrocław, 1992.

[17] A. M. Lister i R. D. Eager. *Wprowadzenie do systemów operacyjnych*. Wydawnictwo Naukowo-Techniczne, Warszawa, 1994.

[18] MIT OpenCourseWare. *Electrical Engineering and Computer Science*. http://aka-ocw.mit.edu/OcwWeb/Global/all-courses.htm.

[19] D. A. Rustling. *The Linux Kernel*. http://www.linuxhq.com/guides/TLK/tlk.html, 1999.

[20] U. Vahalia. *Jądro systemu UNIX®, nowe horyzonty*. Wydawnictwo Naukowo-Techniczne, Warszawa, 2001.

System komputerowy

- użytkownicy (ludzie, maszyny, inne komputery)
- programy użytkowe (kompilatory, edytory, systemy baz danych, gry)
- system operacyjny
- sprzęt (procesor, pamięć, urządzenia wejścia–wyjścia)

System komputerowy wg Tanenbauma

systemy bankowe	rezerwacja biletów	gry
kompilatory	edytory	powłoki
system operacyjny		
język maszynowy		
mikroprogramowanie		
urządzenia fizyczne		

System komputerowy

Poziomy abstrakcji współczesnych systemów komputerowych (wg Null i Lobur)

6	użytkownik	programy wykonywalne
5	język wysokiego poziomy	C, C++, Java, FORTRAN, ...
4	asembler	kod asemblera
3	oprogramowanie systemowe	system operacyjny, biblioteki
2	maszyna	architektura zbioru rozkazów
1	sterowanie	mikrokod lub skonfigurowane sprzętowo
0	logika cyfrowa	obwody, bramki, itd.

Co to jest system operacyjny?

- System operacyjny jest **programem**; działa jako pośrednik między użytkownikiem komputera a sprzętem komputerowym; tworzy środowisko, w którym użytkownik może wykonywać programy.
- System operacyjny **nadzoruje i koordynuje** posługiwanie się sprzętem przez programy użytkowe, które pracują na zlecenie użytkowników.
- System operacyjny jest odpowiedzialny za
  - zarządzanie zasobami komputera
  - tworzenie wirtualnej maszyny (dla programisty)
- **Jądro (*kernel*)** – część systemu operacyjnego, która działa nieustannie; wszystkie pozostałe programy są programami użytkowymi.

Cechy dobrego systemu operacyjnego

- funkcjonalność
- wydajność (procesor powinien pracować dla użytkownika)
- skalowalność (zwiększenie obciążenia i rozbudowa sprzętu)
- niezawodność (dostępność, *five nines*)
- łatwość korzystania i zarządzania

Prehistoria rozwoju komputerów

- Abak, liczydło (znane od około 3000 lat)
- wprowadzenie cyfr arabskich: Gerbert z Aurillac (koniec X w.), Leonardo Fibonacci (przełom XII i XIII w.)
- pałeczki Nepera (wykorzystanie logarytmów): John Neper (Napier), Henry Briggs (początek XVII w.)
- zegar liczący Wilhelma Schickarda: pierwsza maszyna licząca (około 1623)
- Pascaline: arytmometr Blaise'a Pascala (1642)[3]
- programowalne urządzenie włókiennicze: Jaques de Vaucanson (1745)
- pierwsze uniwersalne krosno wykorzystujące karty perforowane: Joseph-Marie Jacquard (pierwsza połowa XIX w.)

[3] Lego Technic – Pascaline #02

Prehistoria rozwoju komputerów

- Maszyna różnicowa Charlesa Babbage'a (około 1823).
  Każdy wielomian może być wyliczany w oparciu o różnice skończone:

$$
\begin{aligned}
f_n &= n^2 + n + 41 \\
\Delta f_n &= f_n - f_{n-1} = 2n \\
\Delta^2 f_n &= \Delta f_n - \Delta f_{n-1} = 2 \\
f_n &= f_{n-1} + \Delta f_n = f_{n-1} + \Delta f_{n-1} + \Delta^2 f_n
\end{aligned}
$$

n	f_n	Δf_n	$\Delta^2 f_n$	$f_{n-1}+\Delta f_{n-1}+\Delta^2 f_n$
0	41	0	2	43
1	43	2	2	47
2	47	4	2	53
3	53	6	2	61
4	61	8	2	71
5	71	10	2	83

Prehistoria rozwoju komputerów

- maszyna różnicowa braci Scheutz (1855): szybkość 33-44 32-cyfrowe liczby na minutę
- maszyna analityczna Babbage'a (1835; Luis Manabrea, Ada Byron)
- *Prawa myślenia* (algebra Boole'a): George Boole (1854)
- karty perforowane jako nośniki informacji: Herman Hollerith (przełom XIX i XX w.)
- pierwszy kalkulator elektroniczny (1939): John Atanasoff, Clifford Berry
- Automatic Sequence Controlled Calculator (ASCC, Harvard Mark I, kalkulator automatycznego sterowania sekwencyjnego): Howard Aiken + inżynierowie IBM (1944) (dodawanie – 0.3 sek, mnożenie – 6 sek)

Generacje komputerów

generacja	lata	technologia	szybkość (operacji/sek)
1	1946-1957	lampa próżniowa	40 000
2	1958-1964	tranzystor	200 000
3	1965-1971	mała i średnia skala scalenia	1 000 000
4	1972-1977	duża skala scalenia	10 000 000
5	1978-	bardzo duża skala scalenia	$>$ 100 000 000

SSI (*Small Scale Integration*) mała skala integracji, 10-100 elem./układ

MSI (*Medium Scale Integration*) średnia skala integracji, 100-1000 elem./układ

LSI (*Large Scale Integration*) duża skala integracji, 1000-10000 elem./układ

VLSI (*Vary Large Scale Integration*) bardzo duża skala integracji, powyżej 10000 elem./układ

Historia rozwoju komputerów i systemów operacyjnych

- 1642-1945 (0. generacja): mechaniczne maszyny liczące
- 1945-1953 (1. generacja): komputery na lampach próżniowych
  - programowanie w języku maszynowym (brak języka programowania)
  - przeprowadzanie wyłącznie obliczeń numerycznych
  - zastosowanie kart dziurkowanych do wprowadzania danych (początek lat 1950.)

ENIAC *Electronic Numerical Integrator and Computer*

Elektroniczny integrator numeryczny i komputer (1946-1955, J.Mauchly, J.P.Eckert):

- 18 tys. lamp próżniowych, 70 tys. oporników, 10 tys. kondensatorów, 1.5 tys. przekaźników, 6 tys. ręcznych przełączników
- 30 ton, 170 m^2, moc 160kW
- 5000 operacji dodawania na sekundę
- maszyna dziesiętna (każda liczba była reprezentowana przez pierścień złożony z 10 lamp)
- ręczne programowanie przez ustawianie przełączników i wtykanie kabli

Park w Bletchley, Anglia, 1939-1945
Government Code and Cipher School, siedziba wywiadu, gdzie łamano niemieckie szyfry przy pomocy ulepszonej „bomby Rejewskiego" (M.Rejewski, J.Różycki, H.Zygalski[4]), a następnie maszyn Heath Robinson, Mark 1 Colossus (1943) i Mark 2 Colossus (1944).

Mark 2 Colossus:

- 2400 lamp elektronowych
- 5 czytników taśm (5×5 tys. znaków na sekundę, bufory)
- arytmetyka binarna
- testowanie Boole'owskich operacji logicznych
- rejestry pamięci elektronicznej sterowane automatycznie
- realizacja podprogramów dla wykonywania określonych funkcji
- wyniki wyprowadzane za pomocą elektrycznej maszyny do pisania

[4] Zob. Otwarcie Centrum Szyfrów Enigma, Centrum Szyfrów w Poznaniu

EDVAC *Electronic Discrete Variable Computer*

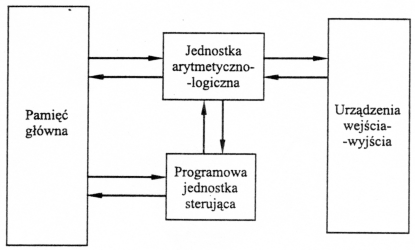
Komputer wg Johna von Neumanna (1946, IAS[5]) (pomysł Mauchly'ego i Eckerta):

- pamięć główna (przechowywanie danych i rozkazów), 1000 słów 40 bitowych; każdy element pamięci ma unikalny identyfikator (zwykle numer) nazywany adresem
- jednostka arytmetyczno-logiczna (ALU, *Arithmetic-Logic Unit*) wykonująca działania na danych binarnych
- jednostka sterująca, która interpretuje rozkazy z pamięci i powoduje ich wykonanie
- urządzenia wejścia-wyjścia, których pracą kieruje jednostka centralna

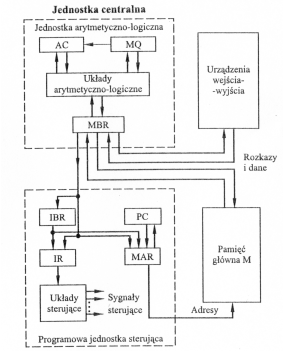
Maszyny von Neumanna – komputery o takiej ogólnej strukturze

[5] Institute for Advanced Study, Princeton

Maszyna von Neumanna



RYSUNEK 2.1. Struktura komputera IAS



RYSUNEK 2.3. Struktura komputera IAS

Jednostka sterująca uruchamia IAS, pobierając rozkaz z pamięci i wykonując go (jeden rozkaz w określonym momencie)

- rejestr buforowy pamięci (MBR – memory buffer register) – przechowywanie słowa, ktore ma być pobrane lub odesłane do pamięci
- rejestr adresowy pamięci (MAR – memory address register) – adres w pamięci słowa, które ma być zapisane/odczytane z MBR
- rejestr rozkazów (IR – instruction register) – 8-bitowy kod operacyjny rozkazu, który jest wykonywany
- buforowy rejestr rozkazów (IBR – instruction buffer register) czasowo przechowuje podręczny rozkaz ze słowa w pamięci
- licznik programu (PC – program counter) – adres kolejnej pary rozkazów
- akumulator (AC – accumulator) oraz rejestr mnożenia i dzielenia (MQ – multiplier-quotient) – czasowe przechowywanie argumentów i wyników operacji prowadzonych przez ALU.

Cykle komputera IAS:

- cykl rozkazu: kod operacji następnego rozkazu jest ładowany do IBR, a część adresowa – do MAR
- cykl wykonania: układy sterujące rozpoznają kod operacji i wykonują rozkaz, wysyłając odpowiednie sygnały sterujące, które powodują, że przenoszone są dane lub ALU wykonuje operację

  Komputer IAS ma 21 rozkazów:

  – przenoszenie danych
  – rozgałęzienie bezwarunkowe
  – rozgałęzienie warunkowe
  – arytmetyka
  – modyfikowanie adresu

Wykonanie operacji mnożenia wymaga wykonania 39 podoperacji.

Lista rozkazów IAS

Kod operacji	Reprezentacja symboliczna	Opis
00001010	LOAD MQ	przenieś zawartość MQ do AC
00001001	LOAD MQ,M(X)	przenieś zawartość komórki X do MQ
00000001	LOAD M(X)	przenieś M(X) do akumulatora
00001101	JUMP M(X,0:19)	pobierz następny rozkaz z lewej połowy M(X)
00001110	JUMP M(X,20:39)	pobierz następny rozkaz z prawej połowy M(X)
00001111	JUMP+M(X,0:19)	jeśli liczba w AC $\geqslant 0$ pobierz nast. rozkaz z lewej połowy M(X)
00001111	JUMP+M(X,20:39)	jeśli liczba w AC $\geqslant 0$ pobierz nast. rozkaz z lewej połowy M(X)
00000101	ADD M(X)	dodaj M(X) do AC i wynik umieść w AC
00000110	SUB M(X)	odejmij M(X) od AC i wynik umieść w AC
00001011	MUL M(X)	pomnóż M(X) przez MQ i umieść najbardziej znaczące bity w AC, a najmniej w MQ
00001100	DIV M(X)	podziel zawartość AC przez M(X), umieść iloraz w MQ, resztę w AC
00010101	LSH	pomnóż AC przez 2 (przesuń w lewo o jedną pozycję)
00010001	RSH	podziel AC przez 2 (przesuń w prawo o jedną pozycję)
00010010	STOR M(X,8:19)	zamień lewe pole adresowe M(X) na 12 bitów AC z prawej

Pierwsze komputery komercyjne

W roku 1947 Mauchly i Eckert tworzą Eckert-Mauchly Computer Corporation. Powstaje pierwszy komputer komercyjny UNIVAC I (*Universal Automatic Computer*), który może realizować

- macierzowe rachunki algebraiczne
- problemy statystyczne
- obliczanie premii dla firm ubezpieczeniowych

UNIVAC I został wykorzystany do powszechnego spisu w 1950 r.

IBM wyprodukował swój pierwszy komputer elektroniczny z przechowywanym programem do zastosowań naukowych w 1953 r. (model 701). Model 702 (1955) znalazł zastosowanie w biznesie.

Historia rozwoju komputerów i systemów operacyjnych

- 1953-1965 (2. generacja): komputery tranzystorowe
  - zwiększona niezawodność
  - rozgraniczenie roli operatora i programisty
  - system wsadowy (wczytywanie i drukowanie *off-line*)
  - karty sterowania zadaniami (*Fortran Monitor System*, *Job Control Language*)

Prosty monitor wraz z językiem kontroli zadań (JCL) umożliwiał przetwarzanie wsadowe (*batch processing*).

Bardziej złożone jestnostki ALU oraz sterujące, ulepszone języki programowania, rozpoczęto dostarczanie **oprogramowania systemowego**.

Rozwój komputerów IBM z serii 700

- zwiększenie pamięci
- skrócenie czasu cyklu pamięci
- wprowadzenie kanałów danych
- zastosowanie multipleksera

Kanał danych: niezależny moduł wejścia-wyjścia z własnym procesorem i własną listą rozkazów.

Multiplekser: szereguje dostęp procesora i kanałów danych do pamięci, urządzenia mogą działać niezależnie.

Historia rozwoju komputerów i systemów operacyjnych

- 1965-1980 (3. generacja): komputery zbudowane z układów scalonych
  - ujednolicenie linii produkcyjnych;
    małe i duże komputery z ujednoliconym systemem operacyjnym (ta sama lista rozkazów, ale różna wielkość pamięci, szybkość procesorów, wydajność)
  - wieloprogramowość, spooling
  - systemy z podziałem czasu: *Compatible Time-Sharing System* (CTSS), *Multiplexed Information and Computing Service* (MULTICS)
  - minikomputery (DEC PDP)

System IBM 360 (1964):

- podobna/identyczna liczba rozkazów
- podobny/identyczny system operacyjny
- rosnąca szybkość
- rosnąca liczba urządzeń we-wy
- rosnący rozmiar pamięci
- rosnąca cena

DEC PDP-8 (1964) – pierwszy minikomputer (magistrala Omnibus)

Historia rozwoju komputerów i systemów operacyjnych

- od 1980 (4. generacja): komputery oparte na układach VLSI
  - układy o dużym stopniu scalenia (VLSI), powstanie stacji roboczych i komputerów osobistych (architektura minikomputerów)
  - sieciowe systemy operacyjne
  - rozproszone systemy operacyjne (układy masywnie równoległe)

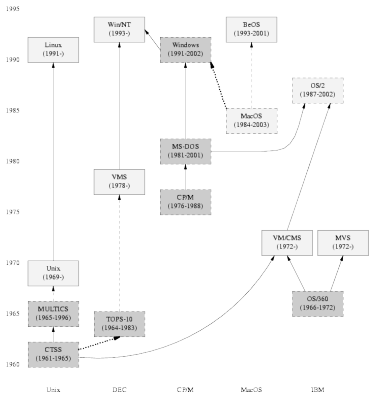
Zalety układów scalonych:

- koszt mikroukładu niezmieniony pomimo wzrostu gęstości upakowania
- gęstsze upakowanie to większa szybkość działania układu
- zmniejszenie wymiarów komputera
- mniejsze zapotrzebowanie na moc i łatwiejsze chłodzenie
- połączenia wewnątrz układu scalonego są bardziej niezawodne, niż połączenia lutowane

Początki komputeryzacji w Polsce

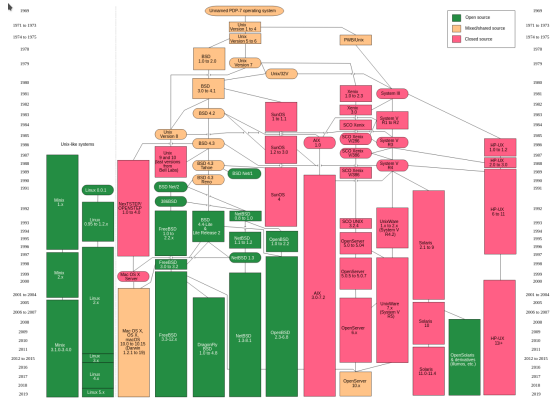
- w grudniu 1948 roku zaczyna działać w Państwowym Instytucie Matematycznym Grupa Aparatów Matematycznych kierowana przez dra Henryka Greniewskiego; dokonania:
  - Analogowy Analizator Równań Algebraicznych Liniowych (ARAL, 1954, K.Bochenek): 400 lamp elektronowych, rozwiązywanie układu co najwyżej 8 (niekoniecznie) liniowych równań różniczkowych.
  - Analogowy Analizator Równań Różniczkowych (ARR, L.Łukaszewicz); Elektroniczna Maszyna Automatycznie Licząca (EMAL, R.Marczyński).
  - XYZ (1958) wzorowany na IBM 701, pamięć naddźwiękową, 800 operacji/sek., przerzutniki z maszyny sowieckiej BESM 6; ZAM 2 (ulepszona wersja XYZ).
- opracowanie Systemu Automatycznego Kodowania (SAKO, 1960), Polski FORTRAN.
- Odra 1002 (1962) – pierwszy komputer z zakładów Elwro, Odra 1003 (1964) – pierwszy seryjnie produkowany komputer (ostatni wyłączono 30/04/2010 po 34 latach pracy)
- 16-bitowy minikomputer K-202 J.Karpińskiego (1970-1973), pierwszy polski komputer zbudowany z użyciem układów scalonych, przewyższał pod względem szybkości pierwsze IBM PC oraz umożliwiał wielozadaniowość, wielodostępność i wieloprocesorowość.
- minikomputer Mera (1978)

Historia systemów operacyjnych wg E.S.Raymonda[6]



[6] *UNIX Sztuka programowania*, Wydawnictwo Helion, Gliwice, 2004. Także http://en.wikipedia.org/wiki/Timeline_of_operating_systems

Historia systemu operacyjnego Unix[7]



[7] Wikipedia: History of Unix

Najważniejsze usprawnienia w organizacji i architekturze komputerów:

- koncepcja rodziny (IBM 360, PDP-8), tzn. oddzielenie architektury od jej realizacji (implementacji)
- mikroprogramowalna jednostka sterująca (IBM linia 360, 1964)
- pamięć podręczna (model 85 IBM S/360, 1968); znaczące poprawienie wydajności
- przetwarzanie potokowe
- zwielokrotnione strumienie (IBM 370/168)
- pobieranie docelowego rozkazu z wyprzedzeniem (IBM 360/91), bufor pętli (CDC 6600, CRAY-1)
- przewidywanie rozgałęzienia (VAX 11/780, 10/1977)
- wieloprocesorowość
- architektura o zredukowanej liczbie rozkazów (RISC)

Ewolucja mikroprocesorów firmy Intel[†]

parametr	8008	8080	8086	80386	80486
rok wprowadzenia	1972	1974	1978	1985	1989
liczba rozkazów	66	111	133	154	235
szerokość szyny adresowej	8	16	20	32	32
szerokość szyny danych	8	8	16	32	32
liczba rejestrów	8	8	16	8	8
adresowalność pamięci	16KB	64KB	1MB	4GB	4GB
szerokość pasma magistrali (MB/s)	-	0.75	5	32	32
czas dodawania rejestr-rejestr (μs)	-	1.3	0.3	0.125	0.06

Mikroprocesory firmy Intel

parametr	286	386	486	Pentium	P6
początek projektowania	1978	1982	1986	1989	1990
rok wprowadzenia	1982	1985	1989	1993	1995
liczba tranzystorów	130K	275K	1.2M	3.1M	5.5M
szybkość (MIPS)	1	5	20	100	150

[†] W.Stallings, *Organizacja i architektura systemu komputerowego*, WNT, Warszawa, 2000.

Prawo Moore'a (1965)

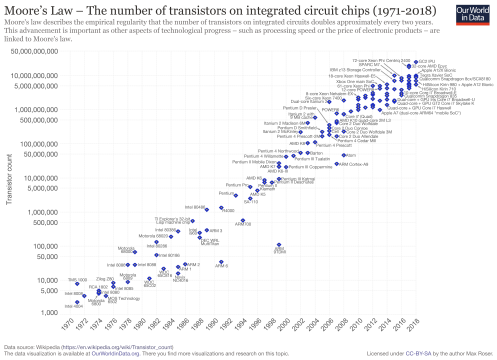
Gordon Moore – założyciel i wiceprezydent firmy Intel

Sformułowanie I:
Liczba tranzystorów, które można zmieścić na jednym calu kwadratowym płytki krzemowej podwaja się co 12 miesięcy.

Sformułowanie II:
Liczba tranzystorów (na jednostce powierzchni płytki krzemowej), która prowadzi do najmniejszych kosztów na jeden tranzystor, podwaja się w przybliżeniu co 12 miesięcy.

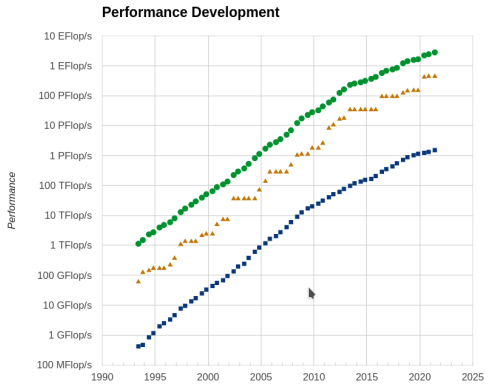
Sformułowanie III:
Wydajność systemów komputerów ulega podwojeniu co około 18 miesięcy.

Prawo Moore'a[8]



[8] https://en.wikipedia.org/wiki/Transistor_count

TOP500: wzrost wydajności superkomputerów

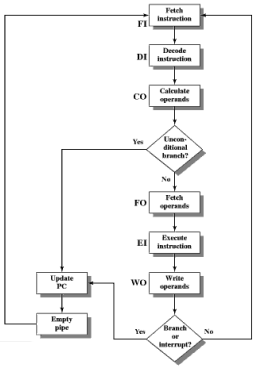


Architektura współczesnego procesora

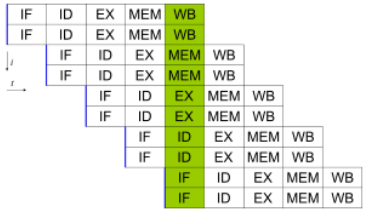
- przetwarzanie potokowe
- superskalarność
- przewidywanie rozgałęzienia (*branch prediction*), spekulatywne wykonywanie rozkazów
- analiza przepływu danych, tj. badanie zależności między rozkazami i wykonywanie ich nawet w kolejności innej niż w programie, aby zmniejszyć opóźnienia
- hiperwątkowość (*hyper-threading*)
- instrukcje SIMD (*Single Instruction Multiple Data*): MMX (*MultiMedia Extensions*), SSE (*Streaming SIMD Extensions*), 3DNow (*3D NO Waiting*)
- wielordzeniowość
- Intel VT (vmx), AMD V (svm)
- GPGPU (*General-Purpose Graphical Processor Unit*)[9]
- Cell-BE (*Cell Broadband Engine*)[10]

[9] CUDA ZONE http://www.nvidia.com/object/cuda_home.html
[10] Roadrunner: PowerXCell 8i 3.2 GHz/Opteron 1.8 GHz, http://www.top500.org

Przetwarzanie rozkazów



Przetwarzanie potokowe i superskalarne[11]



IF=Instruction Fetch, ID=Instruction Decode, EX=Execute, MEM=Memory access, WB=Register write back

[11] http://en.wikipedia.org/wiki/Superscalar, https://en.wikipedia.org/wiki/Instruction_pipelining

Własności procesorów CISC, RISC, superskalarnych (SS)

	CISC		RISC		SS	
	(a)	(b)	(c)	(d)	(e)	(f)
rok powstania	1978	1989	1988	1991	1990	1989
liczba rozkazów	303	235	51	94	184	62
rozmiar rozkazu [B]	2-57	1-11	4	32	4	4,8
tryby adresowania	22	22	3	1	2	11
liczba rejestrów	16	8	32	32	32	23-256
cache [KB]	64	8	16	128	32-64	0.5

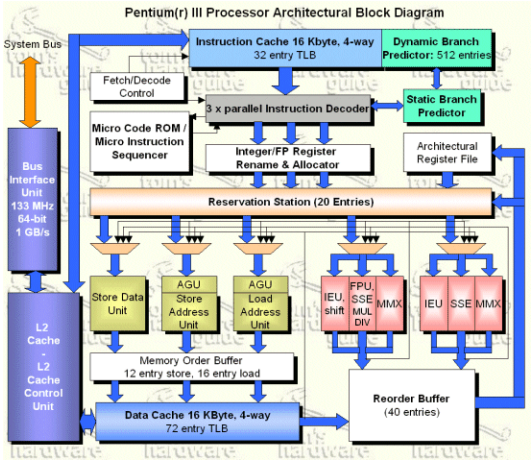
(a) VAX 11/780, (b) Intel 80486, (c) Motorola 88000 (d) MIPS R4000, (e) IBM RS 6000, (f) Intel 80960

CISC (*Complex Instruction Set Computer*) komputer o pełnej liście rozkazów

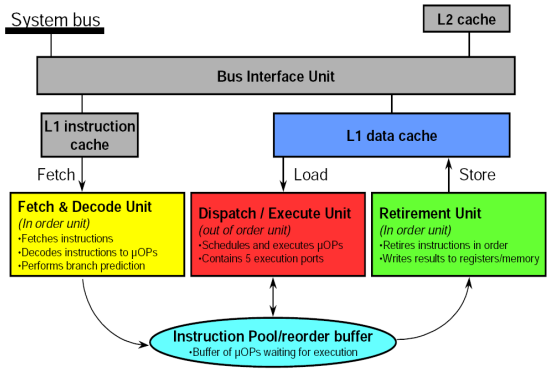
RISC (*Reduced Instruction Set Computer*) komputer o zredukowanej liście rozkazów

Więcej na temat CISC vs RISC i architektury współczesnych procesorów (w tym SoC) w artykułach:

- Erik Engheim: What does RISC and CISC mean in 2020
- Erik Engheim: Why Is Apple's M1 Chip So Fast?

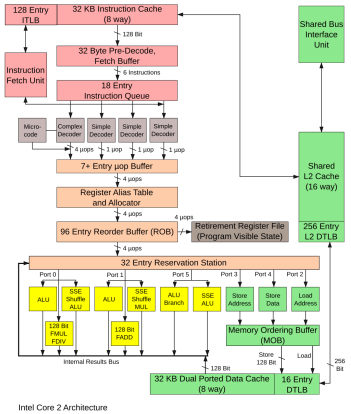


Pentium II and Pentium III Processors Architecture[12]



[12] dynopt.dtc.umn.edu/documents/IA-OPTIMIZATION-Ref-Manual.pdf

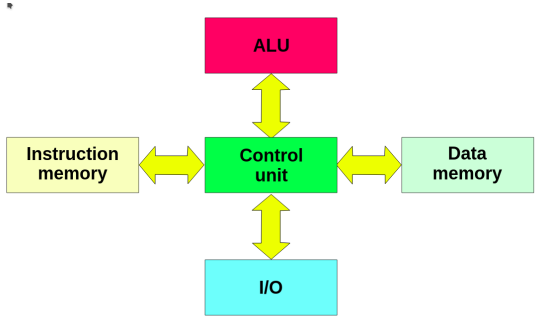
Intel Core 2 Architecture[13]



Intel Core 2 Architecture

[13] https://en.wikipedia.org/wiki/Microarchitecture

Architektura harwardzka[14]



[14] https://en.wikipedia.org/wiki/Harvard_architecture

Pamięć: rodzaje, własności

- szybki rozwój pojemności pamięci i szybkości procesorów
- wolniejszy przyrost szybkości przesyłania danych pomiędzy procesorem i pamięcią

Interfejs pomiędzy pamięcią główną a procesorem jest najbardziej krytycznym elementem całego komputera, ponieważ jest on odpowiedzialny za przepływ rozkazów i danych pomiędzy tymi układami.

Jeśli dostęp do pamięci jest niewystarczający, to cykle procesora są marnowane (głodzenie procesora).

Hierarchia pamięci:

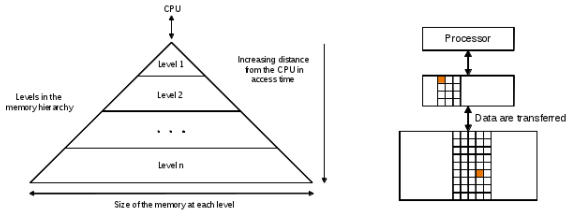
- rejestry (SRAM)
- pamięć podręczna, cache (SRAM)
- pamięć główna (MCDRAM, DRAM, NVRAM, NVDIMM)
- pamięć dyskowa
- pamięć taśmowa, dyski optyczne

Koszt trafienia/chybienia (w cyklach) dla procesora Pentium M:

rejestr	$\leqslant 1$
L1d	≈ 3
L2	≈ 14
pamięć główna	≈ 240

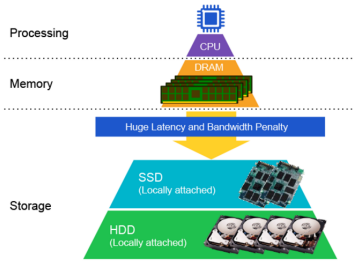
Hierarchia pamięci[15]

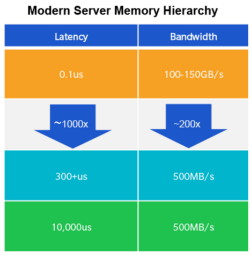
Memory technology	Typical access time	$ per GB in 2004
SRAM	0.5–5 ns	$4000–$10,000
DRAM	50–70 ns	$100–$200
Magnetic disk	5,000,000–20,000,000 ns	$0.50–$2



[15] http://www.site.uottawa.ca/ mbolic/elg6158/caches.ppt

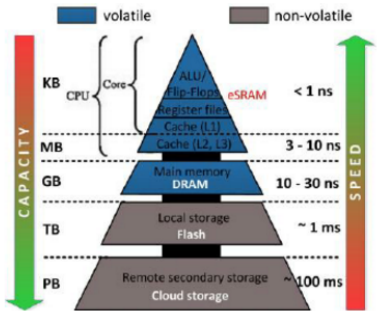
Hierarchia pamięci[16]





[16] https://42xtjqm0qj0382ac91ye9exr-wpengine.netdna-ssl.com/wp-content/uploads/2015/09/server-memory-hierarchy-bandwidth-latency-gap.png

Hierarchia pamięci[17]



[17] https://www.researchgate.net/profile/Bojan_Jovanovic/publication/281805561/figure/fig1/AS:324966131224576@1454489371431/Fig-1-Typical-structure-of-a-computer-memory-hierarchy.png

Hierarchia pamięci[18]

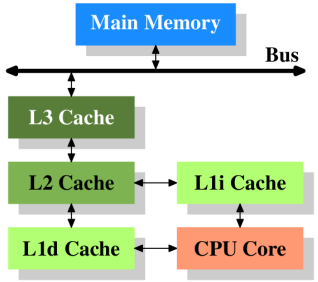
Memory	Size	Latency	Bandwidth
L1 cache	32 KB	1 nanosecond	1 TB/second
L2 cache	256 KB	4 nanoseconds	1 TB/second Sometimes shared by two cores
L3 cache	8 MB or more	10x slower than L2	>400 GB/second
MCDRAM		2x slower than L3	400 GB/second
Main memory on DDR DIMMs	4 GB-1 TB	Similar to MCDRAM	100 GB/second
Main memory on Intel Omni-Path Fabric	Limited only by cost	Depends on distance	Depends on distance and hardware
I/O devices on memory bus	6 TB	100x-1000x slower than memory	25 GB/second
I/O devices on PCIe bus	Limited only by cost	From less than milliseconds to minutes	GB-TB/hour Depends on distance and hardware

[18] https://software.intel.com/en-us/articles/memory-performance-in-a-nutshell

Rodzaj dostępu do pamięci:

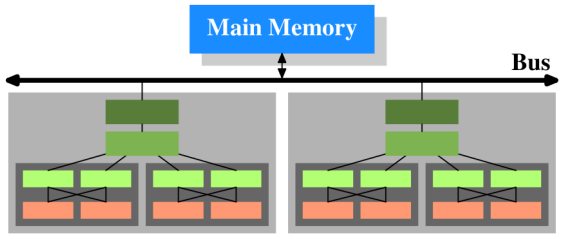
- dostęp sekwencyjny
- dostęp bezpośredni (bloki ew. rekordy mają odrębne adresy)
- dostęp swobodny
- dostęp skojarzeniowy – rodzaj dostępu swobodnego, który umożliwia porównywanie i specyficzne badanie zgodności wybranych bitów wewnątrz słowa (zachodzi to dla wszystkich słów jednocześnie); słowo jest wyprowadzane na podstawie części swojej zawartości, a nie na podstawie adresu (zastosowanie: pamięć podręczna procesora)

Procesor i pamięć[19]



[19] U.Drepper, Red Hat Inc. *What every programmer should know about memory*

Procesor i pamięć[20]



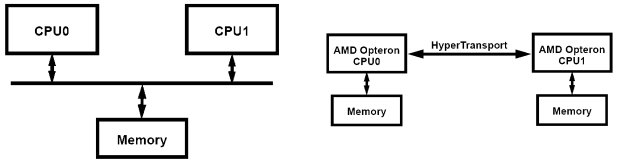
[20] U.Drepper, Red Hat Inc. *What every programmer should know about memory*

Jak organizować pamięć, aby zmniejszyć czas dostępu do danych i instrukcji?

- CPU-cache-magistrala-pamięć (1 słowo)
- CPU-cache-magistrala-pamięć (szeroki dostęp do pamięci)
- CPU-cache-magistrala-(bank 0, bank 1, ...) (pamięć z przeplotem, *interleaved memory*)

  Przykład: Enterprise 6000 z pamięcią 1GB w jednym kawałku był 3-4 razy mniej wydajny niż z pamięcią 4x256MB
- CPU zintegrowana ze sterownikiem pamięci (HyperTransport)
- wydajność magistrali

Procesor i pamięć[21]



[21] http://ixbtlabs.com/articles2/cpu/rmma-numa.html

Wydajność algorytmu (MC)SOR

2. i 4. kolumna – czas w sec., 3. i 5. kolumna – szybkość w MFLOPS/s

system	SOR		MCSOR	
Cray YMP	19.1	(25.7)	3.3	(149)
DEC 8400	5.3	(93.0)	12.1	(40.5)
IBM SP2[a]	7.3	(67.2)	8.5	(57.7)
SGI Power Challenge[b]	6.2	(79.1)	12.1	(40.5)
Sun Enterprise 6000[c]	7.7	(63.7)	15.3	(32.1)
Compaq Proliant 5500[d]	4.3	(114)	9.5	(51.6)
ADAX-Omega[e]	1.0	(491)	1.6	(307)
Dell Power Edge 1750[f]	0.68	(722)	1.30	(377)
PC Pentium IV[g]	0.51	(963)	1.09	(450)
Intel Tiger[h]	0.98	(501)	1.04	(472)
Sun Fire VZ20[i]	0.46	(1067)	1.29	(381)
Sun Fire X4400[j]	0.36	(1380)	0.80	(621)
PC E8400[k]	0.24	(2047)	0.27	(1819)
Dell R610[l]	0.20	(2584)	0.21	(2460)
Dell PowerEdge R530[m]	0.17	(3040)	0.14	(3691)

[a] RS6000/590, 66 MHz wide node
[b] R8000, 90 MHz, 4MB cache
[c] UltraSparc 167 MHz, 0.5 MB cache
[d] Pentium III 550 MHz, Linux Red Hat 8.0, g77
[e] Xeon 1.8 GHz, Linux Red Hat 8.0, g77
[f] Xeon 3.06 GHz, Linux Red Hat 8.0, g77
[g] Prescott 3.0 GHz, Linux, g77
[h] Itanium 2 1.4 GHz, Linux, ifort
[i] Opteron 250 2.4 GHz, Linux (FC3), g77
[j] Opteron 8380 2.5 GHz, Linux CentOS, gfortran
[k] Intel E8400 3.00GHz (6 MB cache), Linux (Fedora9), gfortran
[l] Intel Xeon X5650 2.67GHz, gfortran
[m] Intel Xeon E5-2630 2.40GHz, gfortran

Rodzaje pamięci półprzewodnikowych:[22]

- RAM (*Random Access Memory*) pamięć o dostępie swobodnym – odczyt-zapis, wymazywanie/zapisywanie elektryczne na poziomie bajta
  Static RAM (SRAM), Dynamic RAM (DRAM)
  Synchronous DRAM (SDRAM), Double Data Rate DRAM (DDR)

  $$\text{moc}=\text{pojemność} \times \text{napięcie}^2 \times \text{częstotliwość}$$

  - DDR1: 2.9 V
  - DDR2: 1.8 V, gęstość 2 GB na moduł (2004)
  - DDR3: 1.65 V, gęstość 8 GB na moduł (2007)
  - DDR4: 1.05, 1.2, 1.35 V, gęstość 32 GB na moduł (2014)
- ROM (*Read-Only Memory*) pamięć stała – tylko odczyt, zapisywanie w trakcie wytwarzania (nieużywana)

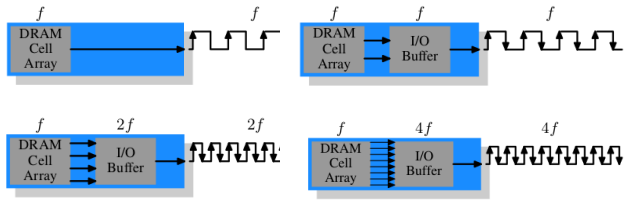
[22] Types of Memory

Rodzaje pamięci półprzewodnikowych

- PROM (*Programmable ROM*) programowalna pamięć stała – tylko odczyt, wymazywanie światłem UV, zapisywanie elektryczne
- EPROM (*Erasable and Programmable ROM*) wymazywalna i programowalna pamięć stała – głównie odczyt, wymazywanie światłem UV, zapisywanie elektryczne
- pamięć błyskawiczna (*flash memory*) – głównie odczyt, wymazywanie elektryczne na poziomie bloku (256 B-16 KB), zapisywanie elektryczne; szybki odczyt, wolniejszy zapis (*Solid-State-Drive*)
- NVRAM (*Non-Volatile RAM*) – pamięć typu SRAM/DRAM z baterią podtrzymującą zawartość po odłączeniu zasilania[23]
- NVDIMM (*Non-Volatile Dual In-line Memory Module*) – pamięć typu DRAM powiązana z układami pamięci błyskawicznej, co pozwala zachować zawartość pamięci ulotnej po awarii zasilania

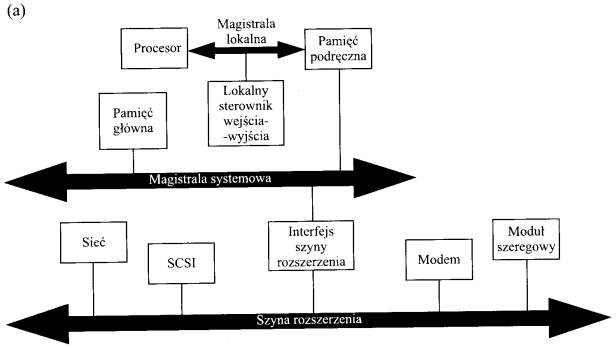
[23] Także EEPROM (*Electrically Erasable and Programmable ROM*) wymazywalna i programowalna pamięć stała (na poziomie bajtu) – głównie odczyt, wymazywanie/zapisywanie elektryczne; szybki odczyt, wolny zapis.

Pamięci DRAM: SDR, DDR1, DDR2, DDR3, DDR4[24]

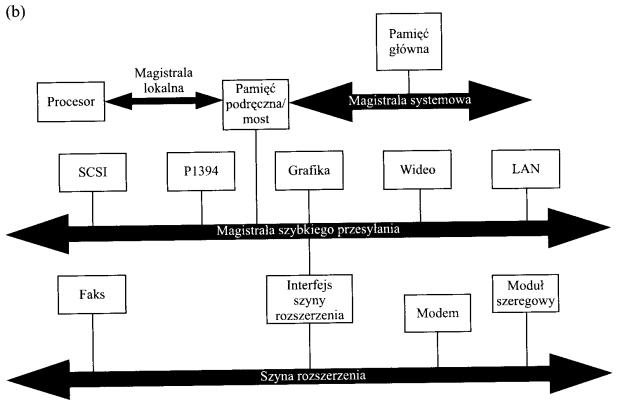


[24] U.Drepper, Red Hat Inc. *What every programmer should know about memory*, DDR1, DDR2, DDR3, DDR4 RAM Differences

Magistrale (szyny) systemowe



Magistrale (szyny) systemowe



Magistrala (E)ISA

(E)ISA (*(Extended) Industry Standard Architecture*) rozszerzona architektura standardu przemysłowego:

- ISA: 8 MHz, szerokość 8-bitów, szybkość 8 MB/s
- EISA: 8 MHz, szerokość 32-bity, szybkość 33 MB/s

Magistrala PCI

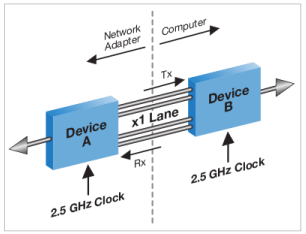
PCI (*Peripheral Component Interconnect*) interfejs komponentów peryferyjnych:

- 32/64-bitowa szyna rozszerzeń dla komputerów zgodnych z IBM PC oraz Macintosh
- opracowana przez firmę Intel w 1992 r. (mikroprocesor Pentium)
- obsługuje standard podłącz i używaj (PnP, *Plug and Play*)
- szybkość transferu danych (MB/s=10^6 B/s, GB/s=10^9 B/s)

PCI32	33.33 MHz =	133.3 MB/s	66.66MHz = 266.6 MB/s
PCI64	33.33 MHz =	266.6 MB/s	66.66MHz = 533.3 MB/s
PCI-X	133.32 MHz =	1066.4 MB/s	266.64MHz = 2133.1 MB/s
PCIe 1.0 (×1)	2.50 GHz	250.0 MB/s	
PCIe 1.0 (×16)		4.0 GB/s	
PCIe 2.0 (×16)	5.00 GHz	8.0 GB/s	
PCIe 3.0 (×16)	8.00 GHz	15.754 GB/s	
PCIe 4.0 (×16)	16.00 GHz	31.508 GB/s	

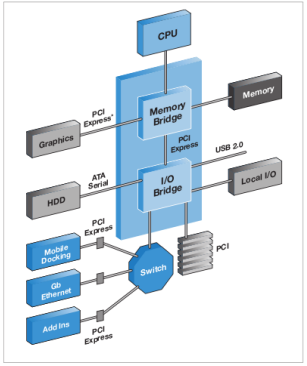
Magistrala PCI Express

Figure 3. PCI Express* x1 lane. A lane consists of two differentially driven signal pairs between two PCI Express devices (Device A and Device B).



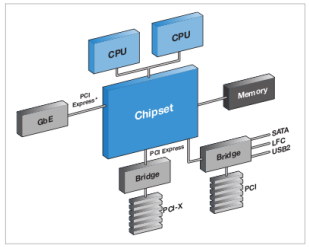
Magistrala PCI Express: płyta główna PC

Figure 4. Gigabit Ethernet to the desktop implemented through the high-speed serial I/O bus of PCI Express*.



Magistrala PCI Express: płyta główna serwera

Figure 5. Example of PCI Express* in server/workstation systems. The PCIe* switch is integrated into the chipset.



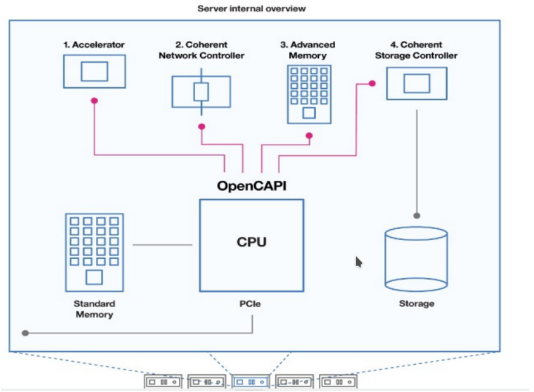
Magistrala przyszłości – OpenCAPI (2016-10-15)

IBM has unveiled OpenCAPI, an open-standard, high-speed bus interface for connecting devices in servers. The announcement coincides with the formation of a consortium of the same name that will manage the new standard, and which initially includes tech heavyweights Hewlett Packard Enterprise (HPE), Dell EMC, NVIDIA, Mellanox, Micron, Xilinx, and Google. The first OpenCAPI-supported devices and servers are expected to show up in 2017.

Unlike CAPI (*Coherent Accelerator Processor Interface*), IBM's coherent memory communication protocol that rode on top of PCI Express (PCIe), OpenCAPI is both a new hardware interface and a new protocol, and would obviate the need for PCIe on the motherboard if all the chips were OpenCAPI compliant. The general idea is the same as IBM's original CAPI: to be able to link together processors, coprocessors network adapters, flash modules, and other devices in the server as equal peers, and which can directly access system memory in the same manner as a CPU.

Zob. New Standard for High Performance Server Bus

Heterogeniczna architektura OpenCAPI[25]



[25] OpenCAPI: a new standard for high performance attachment of memory acceleration and networks

Uniwersalna szyna szeregowa (*Universal Serial Bus*, USB):

- standard szyny zewnętrznej do podłączania do komputera do 127 urządzeń peryferyjnych (jeden IRQ)
- standard opracowany w 1995 r. wspólnie przez wiodących producentów sprzętu komputerowego i telekomunikacyjnego (Compaq, DEC, IBM, Intel, Microsoft, NEC, Northern Telecom, Philips)
- szybkość: 12 Mb/s i 1.5 Mb/s (USB 1.1), 480 Mb/s (USB 2.0), 4.8 Gb/s (USB 3.0), 10 Gb/s (USB 3.1), 20 Gb/s (USB 3.2), 40 Gb/s (USB 4.0)
- długość kabla: USB 1.1 – 3 m, USB 2.0 – 5 m , USB 3.0/3.1 – 3 m
- transmisja w trybie półdupleksu (USB 1.1 i 2.0) i pełnego dupleksu (USB 3.0)
- wyprowadzenie funkcji PnP poza komputer
- łatwość rozmnażania portów i wydłużania połączenia poprzez zastosowanie maksymalnie pięciu koncentratorów (*USB hubs*)
- USB łączy drukarki, skanery, kamery wideo, dyski, stacje dyskietek, klawiatury, myszy, joysticki, telefony, modemy, napędy CD-ROM, napędy taśmowe, urządzenia wideo MPEG-2, *data gloves*, digitalizatory (*digitizers*), itp.

Interfejsy dysków: IDE/ATA

(E)IDE (*(Extended)Intelligent/Integrated Drive Electronics*) właściwie ATA (*AT Attachement*), od 2003 – PATA (*Parallel ATA*)

- EIDE – napędy IDE z lat 1994-1997
- UltraDMA (DMA-33, Ultra33, ATA-33)– napędy IDE z lat 1997-1999 (16 MB/s), najnowsze napędy CD-ROM and CD-RW
- ATA-66 (Ultra66, DMA-66) – napędy IDE z lat 1999-2000
- ATA-100, ATA-133 – szybkość transferu do 133 MB/s (teoret.), $\approx$ 20 MB/s (prakt.)
- ATAPI – *ATA Packet Interface* dla napędów CD, DVD, ...
- MTBF $\approx 1.2 \times 10^6$ h[26]

Ograniczenia: każdy sterownik IDE potrzebuje IRQ i może obsłużyć 2 urządzenia wewnętrzne (*master* i *slave*); maks. dł. kabla – 475 mm.

[26] MTBF (*Mean Time Between Failures*) – czas po jakim z pewnej puli urządzeń zostaje połowa sprawnych. Z 1000 dysków o MTBF 1.2×10^6 h będzie się psuł jeden co (średnio) 50 dni.

Interfejsy dysków: SATA

Szeregowy ATA (serial ATA, SATA) – najnowsza wersja IDE

- szybkość (teor.): 150 MB/s (1. generacja), 300 MB/s (2. generacja), 600 MB/s (docelowa)
- szybkość (prakt.): $\approx$ 40 MB/s (SATA 2), 170 MB/s (SATA 3)
- połączenia punkt-punkt
- możliwość podłączania wewnętrzych i zewnętrznych urządzeń
- CRC (*Cyclic Redundancy Check*) dla pakietów danych, rozkazów i statusu
- łatwość przyłączenia (wsparcie dla *hot swapping*)
- MTBF $\approx 1.2 \times 10^6$ h

Dyski SSD (*Solid State Drive*)

- Problem skończonej liczby cykli P/E
  Żywotność współczesnych dysków jest rzędu kilkuset lat!
- SATA III 6Gbps
  - szybkość 460-560 MB/s[27]
- PCIe 4.0
  - szybkość 2500-7000 MB/s
- NVMe
  - szybkość 3000-3500 MB/s

[27] Wyznaczanie szybkości operacji dyskowych: `hdparm -tT /dev/sda`.

SCSI (*Small Computer Systems Interface*)

- jeden IRQ, obsługa do 16 urządzeń (w tym sterownik), łącze wielopunktowe (*multidrop*) dyski, napędy CD-ROM, napędy taśmowe (*streamer*), skanery
- SCSI: 3 MB/s
- Fast SCSI: 8 MB/s
- Fast & Wide SCSI: 16 MB/s
- Ultra 160: 30-60 MB/s
- Ultra 320: 60-120 MB/s
- MTBF $\approx 1.6 \times 10^6$ h

Urządzenia EIDE i UltraDMA bardziej obciążają CPU niż urządzenia SCSI.

SCSI: rozwój standardu[28]

STA Terms	Clock Speed (MHz)	Bus Speed (MB/s)	Max Bus Width (bits)	Max. Bus Lengths(m) SE	LVD	HVD	Max. Dev. Supp.
SCSI-1	5	5	8	6		25	8
Wide SCSI	5	10	16	3		25	8
Fast SCSI	10	10	8	3		25	8
Fast Wide SCSI	10	20	16	3		25	16
Ultra SCSI	20	20	8	1.5		25	8
Ultra SCSI	20	20	8	3			4
Wide Ultra SCSI	20	40	16			25	16
Wide Ultra SCSI	20	40	16	1.5			8
Wide Ultra SCSI	20	40	16	3			4
Ultra2 SCSI	40	40	8		12	25	8
Wide Ultra2 SCSI	40	80	16		12	25	16
Ultra-160 SCSI	80	160	16		12		16
Ultra-320 SCSI	160	320	16		12		16

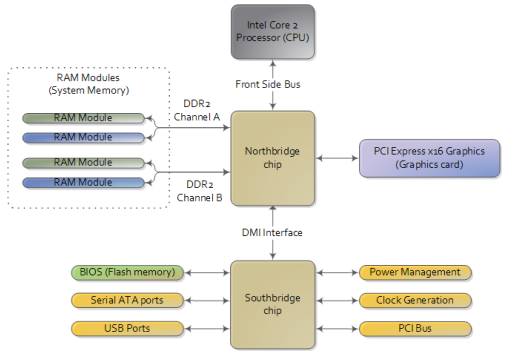
STA – SCSI Trade Association

[28] http://www.scsita.org/aboutscsi/SCSI_Termination_Tutorial.html

SAS (*Serial Attached SCSI*)

- jeden IRQ, połączenie punkt-punkt (*initiator-target*), tryb pełnego dupleksu
- zastosowanie ekspanderów (*expander*) daje możliwość podłączenia do 65535 urządzeń
- szybkość 3.0, 6.0, 12.0, 22.5 Gb/s (45 Gb/s) $\approx$ 90 MB/s (prakt.)
- MTBF $\approx 1.6 \times 10^6$ h

Architektura płyty głównej[29]



[29] G. Duarte http://duartes.org/gustavo/blog/post/motherboard-chipsets-memory-map

Jak system wykonuje zadania?

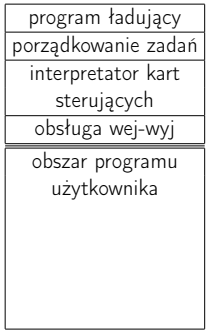
- Monitor prosty
- Praca pośrednia
- Buforowanie
- Spooling
- Wieloprogramowość
- Systemy z podziałem czasu

Monitor prosty

W czasie instalowania taśm jednostka centralna była bezczynna. Komputery były drogie i ich czas był cenny.

Rozwiązanie:

- zatrudnienie profesjonalnych operatorów
- przetwarzanie wsadowe (*batch processing*)
- automatyczne porządkowanie zadań (*automatic job sequencing*)
- monitor rezydujący – automatyczne przekazywanie sterowania od zadania do zadania; karty sterowania zadaniami (JCL, *Job Control Language*)



Monitor rezydujący w pamięci

Praca pośrednia

- Wolne operacje wej-wyj wykonywane przez mniejsze, satelitarne minikomputery.
- Jednostka główna czyta dane z taśmy magnetycznej i umieszcza wyniki na taśmie magnetycznej.
- Obsługa czytników kart i drukarek wierszowych w trybie pośrednim (*off-line*).
- Niezależność od urządzeń wej-wyj – programy pisane z myślą o korzystaniu z logicznych, a nie fizycznych urządzeń peryferyjnych.

Buforowanie

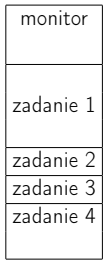
- Buforowanie jest metodą jednoczesnego wykonywania obliczeń i operacji wej-wyj dla danego zadania.
- Zadania uzależnione od wej-wyj (*I/O bound*) – prędkość przetwarzania zależna od szybkości urządzeń wej-wyj.
- Zadania uzależnione od jednostki centralnej (*CPU bound*) – bufor wejściowy zawsze pełny, bufor wyjściowy zawsze pusty.

Spooling

- spooling (*Simultaneous Peripheral Operation On-Line*) jednoczesna bezpośrednia praca urządzeń
- spooling jest stał się możliwy, gdy rolę taśm magnetycznych przy przetwarzaniu pośrednim zastąpiły dyski
- spooling umożliwia wykonywanie operacji wej-wyj jednego zadania i obliczeń dla innych zadań
- spooling wytwarza pulę zadań do wykonania i umożliwia planowanie zadań i wieloprogramowość

Wieloprogramowość

- Praca pośrednia, buforowanie, spooling mają swoje ograniczenia. Jeden użytkownik nie jest w stanie angażować stale urządzeń wej-wyj i jednostki centralnej!

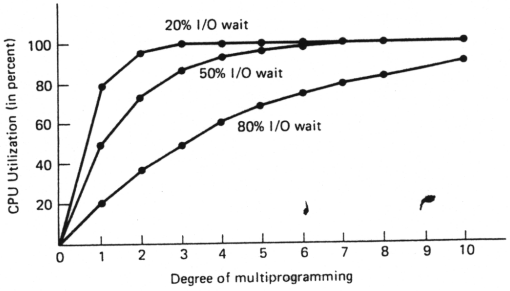


- Realizacja wieloprogramowości wymaga skomplikowanego systemu operacyjnego: ochrona zadań między sobą i planowanie przydziału procesora, kolejki, partycje o stałych lub zmiennych wielkościach

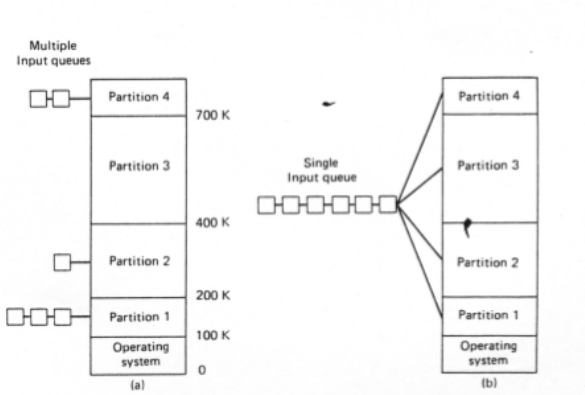
Wieloprogramowość: wykorzystanie CPU

- Prosty model wykorzystania CPU:
  - n-procesów, każdy spędza część p swego czasu w stanie I/O
  - p^n – prawdopodobieństwo, że n-procesów będzie w stanie I/O
  - wykorzystanie CPU $= 1 - p^n$
- Znaczenie pamięci w wydajnym funkcjonowaniu współczesnych systemów operacyjnych.

Wieloprogramowość: wykorzystanie CPU (cd)



Wieloprogramowość: wykorzystanie pamięci (cd)



Systemy z podziałem czasu

- praca wsadowa a praca interakcyjna
- wieloprogramowość $\longrightarrow$ wielozadaniowość (=podział czasu)
- system z podziałem czasu – wielu użytkowników dzieli jeden komputer
- systemy z podziałem czasu są skomplikowane, gdyż wymagają:
  - realizacji mechanizmów działań współbieżnych
  - zarządzania pamięcią
  - ochrony pamięci
  - planowania przydziału CPU
  - administrowania pamięcią dyskową
  - systemu plików dostępnych bezpośrednio

Podział systemów HPC (*High Performance Computing*)

- SMP (*Symmetric MultiProcessing*)
- DSM (*Distributed Shared Memory*)
- MPP (*Massively Parallel Processor*), rozproszona pamięć
- klastry obliczeniowe
- konstelacje obliczeniowe

Zalety systemów wieloprocesorowych

- podział zasobów
- przyspieszenie obliczeń
- niezawodność
- komunikacja

Rygorystyczne systemy czasu rzeczywistego

- sterowniki urządzeń o ścisłe określonym zastosowaniu: nadzorowanie procesów produkcyjnych, eksperymentów naukowych, kierowanie sygnalizacją świetlną, autopilot, itp.
- działanie podlega ostrym rygorom czasowym

Łagodne systemy czasu rzeczywistego: działanie podlega złagodzonym rygorom czasowym. Zastosowania: techniki multimedialne, tworzenie wirtualnej rzeczywistości, urządzenia zdolne do samodzielnej eksploracji.

Systemy z obsługą przerwań

Jak ma współdziałać jednostka centralna z urządzeniami wej-wyj?

- aktywne czekanie
- odpytywanie
- przerwania
- bezpośredni dostęp do pamięci

Aktywne czekanie

1. sprawdź czy drukarka jest gotowa na przyjęcie następnego znaku
2. jeśli nie jest gotowa, to idź do punktu 1
3. jeśli drukarka jest gotowa (po wydrukowaniu znaku), to sprawdź czy jest do wydrukowania nowy znak
4. jeśli jest nowy znak, to idź do punktu 1
5. jeśli nie ma więcej znaków, to drukowanie zostało zakończone

Odpytywanie (*polling*)

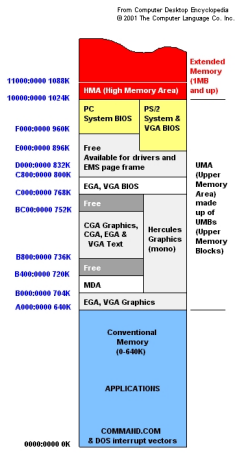
1. wybierz kolejne urządzenie wymagające obsługi
2. sprawdź, czy to urządzenie wymaga obsługi
3. jeśli tak, to uruchom procedurę obsługi urządzenia
4. jeśli nie, to przejdź do punktu 1.

Przerwania

- Sterownik urządzenia (*device controller*) związany jest z konkretnym urządzeniem i rozporządza lokalnym buforem i zbiorem rejestrów o specjalnym przeznaczeniu. Odpowiada za przesyłanie danych między urządzeniem zewnętrznym, a własnym buforem.
- Moduł sterujący/obsługi urządzenia (*driver*) jest odpowiedzialny od strony systemu operacyjnego za komunikację ze sterownikiem urządzenia.

```
00:1f.3 SMBus: Intel Corporation 8 Series SMBus Controller (rev 04)
        Subsystem: Lenovo Device 220c
        Flags: medium devsel, IRQ 18
        Memory at e0638000 (64-bit, non-prefetchable) [size=256]
        I/O ports at efa0 [size=32]
        Kernel driver in use: i801_smbus
        Kernel modules: i2c_i801
```

Zob.: `/proc/iomem`, `/proc/ioports`




```
          0.0000 M          0.0039 M  Reserved
          0.0039 M          0.6133 M  System RAM
          0.6133 M          0.6250 M  Reserved
          0.6250 M          0.7500 M  PCI Bus 0000:00
          0.7500 M          0.8076 M  Video ROM
          0.8125 M          0.8281 M  pnp 00:00
          0.8281 M          0.8437 M  pnp 00:00
          0.8438 M          0.8594 M  pnp 00:00
          0.8594 M          0.8750 M  pnp 00:00
          0.8750 M          1.0000 M  Reserved
          0.9375 M          1.0000 M  System ROM
          1.0000 M       2785.2656 M  System RAM
        352.0000 M        360.6260 M  Kernel code
        360.6260 M        368.3481 M  Kernel data
        370.6484 M        372.5781 M  Kernel bss
       2785.2656 M       3000.4531 M  Reserved
       3000.4531 M       3000.7656 M  ACPI Non-volatile Storage
       3000.7656 M       3020.4648 M  Reserved
       3020.4648 M       3022.4961 M  ACPI Non-volatile Storage
       3022.4961 M       3022.9961 M  ACPI Tables
...
       4077.2500 M       4077.2969 M  PCI Bus 0000:00
       4077.2500 M       4077.2695 M  TPM
       4077.2695 M       4077.2969 M  pnp 00:01
       4078.0000 M       4078.0039 M  Local APIC
       4078.0000 M       4078.0039 M  Reserved
       4092.0000 M       4096.0000 M  Reserved
```


Przerwania: działanie

- jednostka centralna inicjuje przesyłanie danych przez wprowadzenie pewnych wartości do odpowiednich rejestrów sterownika urządzenia
- sterownik urządzenia rozpoczyna działanie (gromadzenie danych w buforze)
- sterownik urządzenia powiadamia procesor o zakończonej pracy generując określone przerwanie
- procesor wstrzymuje bieżącą pracę, odkłada na stos adres przerwanego rozkazu, określa źródło przerwania i przekazuje sterowanie do procedury obsługi przerwania *(interrupt handler, interrupt service routine (ISR))* wykorzystując wektor przerwań lub odpytywanie
- przesłanie danych z bufora do programu użytkownika
- wznowienie przerwanej pracy

Procedura obsługi przerwania jest częścią modułu obsługi urządzenia, czyli kodu jądra zarządzającego urządzeniem.

Przerwania: obsługa[30]

- przerwanie sprzętowe może zostać zainicjowane w dowolnym momencie
- wywołanie procedury obsługi przerwania zachodzi w dowolnym momencie; procedura obsługi przerwania powinna zostać uruchomiona jak najszybciej
  - wymagania urządzenia: jak najszybsza obsługa przerwania
  - wymagania systemu operacyjnego: jak najkrótsza obsługa
    - górna połówka – etap 1. uruchamiany bezpośrednio po odbiorze sygnału przerwania, realizuje tylko te czynności, które mają kluczowe znaczenie dla funkcjonowania urządzenia (przyjęcie przerwania, restart urządzenia)
    - dolna połówka – etap 2. obejmuje czynności obsługi, które mogą być odłożone w czasie (po wyłączeniu innych przerwań)
  - szeregownie wywołania procedur obsługi przerwań (*ksoftirq*)

[30] https://www.supportsages.com/what-does-ksoftirqd-do/
http://www.jonmasters.org/blog/2007/12/12/everything-you-know-about-interrupts-is-wrong/

Przerwania: podział

- asynchroniczne (przerwania) – generowane przez inne urządzenia sprzętowe w dowolnym czasie, niezależnie od sygnałów zegarowych procesora.
- synchroniczne (wyjątki) – tworzone przez jednostkę sterowania procesora podczas wykonywania instrukcji (są synchroniczne, bo generowane są po zakończeniu wykonywania instrukcji)

Przerwania: źródła

Systemy z obsługą przerwań są kierowane zdarzeniami generowanymi przez

- sprzęt (zegar i urządzenia I/O) – przerwania
- oprogramowanie (błędy programisty, nieprawidłowe operacje) – wyjątki, pułapki

Przerwania: podział wg dokumentacji Intela

Przerwania sprzętowe:

- IRQ – *Interrupt ReQuests* generowane przez urządzenia (przerwania maskowalne); pin INTR procesora
- niemaskowalne – krytyczne zdarzenia związane z awarią sprzętu; pin NMI (*NonMaskable Interrupt*)
- IPI – *Inter-Processor Interrupt* (tylko dla SMP)

Przerwania: podział wg dokumentacji Intela

Wyjątki:

- wyjątki wykrywane przez procesor – generowane, gdy proces wykryje nieprawidłowości przy wykonywaniu instrukcji (IEEE 754: niedomiar (*underflow*), nadmiar (*overflow*), dzielenie, niepoprawna operacja, niedokładna operacja, dzielenie przez zero
- błędy (*faults*) – instrukcja, która spowodowała błąd może być wznowiona
- pułapki (*traps*) – licznik instrukcji jest ustawiany na adres kolejnej instrukcji (śledzenie programu)
- załamania (*aborts*) – wystąpił poważny błąd i jednostka sterowania nie może zachować adresu instrukcji; zwykle oznacza zakończenie procesu, który spowodował załamanie
- wyjątki programowe (przerwania programowe) – występują na żądanie programisty

Przerwania: wady

- szybkość transferu wej-wyj jest ograniczona szybkością, z jaką procesor może testować i obsługiwać urządzenie
- procesor jest zajęty zarządzaniem przesyłania z wejścia i na wyjście (wykonywane są specjalne rozkazy)

Szybkie urządzenia wej-wyj oraz urządzenia przesyłające duże ilości danych wymagają bezpośredniego dostępu do pamięci (DMA).

Bezpośredni dostęp do pamięci wymaga dodatkowego modułu na magistrali systemowej (moduł DMAC, *DMA Controller*, część mostka południowego).

Przerwania: DMA

Transfer danych wymaga przekazania przez procesor do modułu DMA rozkazu zawierającego informacje:

- czy wymagany jest odczyt/zapis
- adres urządzenia wej-wyj
- adres początkowej komórki pamięci z danymi (na dane)
- liczbę słów do przesłania

Procesor inicjuje przesłanie danych i kontynunuje przetwarzanie do momentu nadejścia przerwania od modułu DMAC.

Sterowanie zdarzeniami (przerwaniami) rodzi problem synchronizacji w dostępie do zasobów.

Przerwania: identyfikacja

Każde przerwanie lub wyjątek jest identyfikowane przez liczbę z zakresu od 0 do 255 (Intel nazywa tę 8-bitową liczbę bez znaku *wektorem*). Wektory przerwań niemaskowalnych i wyjątków są stałe.

- Wektory 0-31 odpowiadają wyjątkom i przerwaniom niemaskowalnym.
- Wektory 32-255 definiowane przez użytkownika

  Linux używa tylko wektora 128, czyli 0x80, za pomocą którego realizowane są wywołania systemowe. Instrukcja asemblera int 0x80 powoduje przełączenie procesora w tryb jądra i zaczyna się wykonywanie funkcji jądra system_call().

Przyporządkowanie wyjątków do sygnałów

#	Wyjątek	Sygnał
0	*Divide error*	SIGFPE
1	*Debug*	SIGTRAP
2	NMI	
3	*Breakpoint*	SIGTRAP
4	*Overflow*	SIGSEGV
5	*Bounds check*	SIGSEGV
6	*Invalid opcode*	SIGILL
7	*Device not available*	SIGSEGV
8	*Double fault*	SIGSEGV
9	*Coprocessor segment overrun*	SIGFPE
10	*Invalid TSS*	SIGSEGV
11	*Segment not present*	SIGBUS
12	*Stack exception*	SIGBUS
13	*General protection*	SIGSEGV
14	*Page fault*	SIGSEGV
15	zarezerowana przez Intela	
16	*Floating point error*	SIGFPE
17	*Alignment check*	SIGSEGV

Przyporządkowanie IRQ do urządzeń I/O (chip 8259A *master+slave*)

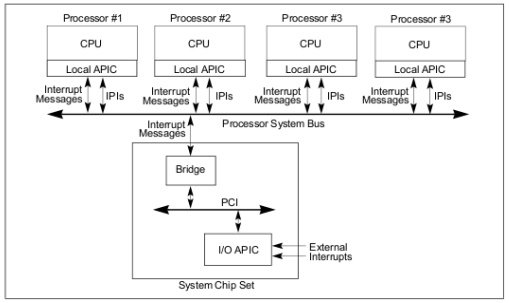
IRQ	INT	urządzenie
0	32	zegar
1	33	klawiatura
2	34	kaskada PIC
3	35	drugi port szeregowy
4	36	pierwszy port szeregowy
5	37	karta dźwiękowa
6	38	stacja dysków
7	39	port równoległy
8	40	zegar systemowy
11	43	interfejs sieciowy
12	44	mysz PS/2
13	45	koprocesor matematyczny
14	46	pierwszy łańcuch sterownik dysków EIDE
15	47	drugi łańcuch sterownika dysków EIDE

Przerwania: sterownik przerwań

Sterownik urządzenia sprzętowego, generuje przerwanie korzystając z linii IRQ (*Interrupt ReQuest*, żadanie przerwania). Wszystkie linie IRQ są podłączone do pinów wejściowych układu zwanego programowalnym sterownikiem przerwań (*Programmable Interrupt Controller*, PIC).
PIC wykonuje następujące zadania:

1. monitoruje linie IRQ sprawdzając podnoszone sygnały
2. jeżeli na linii IRQ zdarzy się podniesienie sygnału:
   (a) tłumaczy otrzymany sygnał na odpowiedni wektor
   (b) zachowuje wektor w porcie I/O sterownika przerwań, co pozwala na odczytanie go przez procesor za pomocą szyny danych
   (c) wysyła sygnał do pinu INTR procesora (generuje przerwanie)
   (d) czeka, aż procesor potwierdzi otrzymanie sygnału przerwania poprzez dokonanie wpisu do jednego z portów sterownika przerwań
3. wraca do punktu 1

APIC dla procesorów Intel Xeon w systemach SMP[31]



[31] Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1

Przyporządkowywanie przerwań

```
$ less /var/log/messages

Sep 22 09: ... kernel: ttyS00 at 0x03f8 (irq = 4) is a 16550A
Sep 22 09: ... kernel: PCI: Found IRQ 5 for device 00:08.0
Sep 22 09: ... kernel: maestro: Configuring ESS Maestro 2E found at IO 0xD800 IRQ 5
Sep 22 09: ... kernel: parport0: irq 7 detected
Sep 22 09: ... kernel: lp0: using parport0 (polling).
Sep 22 09: ... kernel: ttyS04 at port 0x4880 (irq = 11) is a 16550A
Sep 22 22: ... kernel: eth0: Xircom Cardbus Adapter rev 3 at 0x4800, 00:10:A4:D2:52:55, IRQ 11.
Sep 22 09: ... kernel: ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
Sep 22 09: ... kernel: ide1 at 0x170-0x177,0x376 on irq 15
```


Monitorowanie przerwań sprzętowych

```
# uname -sr
Linux 2.6.18-92.1.10.el5 x86_64+

# cat /proc/interrupts
           CPU0       CPU1
  0:  799260896          0    IO-APIC-edge  timer
  1:        140        100    IO-APIC-edge  i8042
  6:          5          0    IO-APIC-edge  floppy
  8:          0          0    IO-APIC-edge  rtc
  9:          0          0    IO-APIC-level  acpi
 12:        115          0    IO-APIC-edge  i8042
 15:    7173047       3915    IO-APIC-edge  ide1
169:          0          0    IO-APIC-level  ohci_hcd:usb1, ohci_hcd:usb2
177:     567795     125000    IO-APIC-level  ioc0
185:    6118456      40533    IO-APIC-level  eth0
NMI:       2770       1163
LOC:  799176280  799176233
ERR:          0
MIS:          0
```


Monitorowanie przerwań sprzętowych

```
# uname -sr
Linux 3.16.7-200.fc20.x86_64+
# cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
   0:         52          0          0          0   IO-APIC-edge      timer
   1:    1006330       4440        296        437   IO-APIC-edge      i8042
   8:        148          1          0          0   IO-APIC-edge      rtc0
   9:    2146357      13230       2323       2883   IO-APIC-fasteoi   acpi
  12:    7693889     262201      41589      42615   IO-APIC-edge      i8042
  18:          0          0          0          0   IO-APIC-fasteoi   i801_smbus
  23:       2678         31          7          0   IO-APIC-fasteoi   ehci_hcd:usb1
  56:   23028438       7450       2085     270885   PCI-MSI-edge      ahci
  57:         16          0          0          0   PCI-MSI-edge      mei_me
  58:         13          2          0          0   PCI-MSI-edge      snd_hda_intel
  59:        170   13483586          3          4   PCI-MSI-edge      i915
  60:        436         42          0          0   PCI-MSI-edge      snd_hda_intel
  61:      49675         19          0          0   PCI-MSI-edge      rtsx_pci
  62:       5024      16752          0          0   PCI-MSI-edge      xhci_hcd
  63:    8625428         70         35         27   PCI-MSI-edge      iwlwifi
  64:     126711    1055448          0          0   PCI-MSI-edge      em1
 NMI:          0          0          0          0   Non-maskable interrupts
 LOC:  215494306  111932313  169406908  101776195   Local timer interrupts
 SPU:          0          0          0          0   Spurious interrupts
 PMI:          0          0          0          0   Performance monitoring interrupts
 IWI:          7          2          0          0   IRQ work interrupts
 RTR:        135          1          0          0   APIC ICR read retries
 RES:    3739391    3236592    2903216    2257333   Rescheduling interrupts
 CAL:      51504      59840      55088      62537   Function call interrupts
 TLB:     708240     650891     700244     664631   TLB shootdowns
 TRM:          0          0          0          0   Thermal event interrupts
 THR:          4          4          0          0   Threshold APIC interrupts
 MCE:          0          0          0          0   Machine check exceptions
 MCP:       1904       1874       1870       1870   Machine check polls
```


Monitorowanie przerwań sprzętowych: klawiatura

Działanie: naciśnięcie i przytrzymanie pojedynczego klawisza

```
root@nscobie ~$ sar -I 1 1
Linux 5.14.16-101.fc33.x86_64 (nscobie)         15.11.2021      _x86_64_        (8 CPU)

09:38:15          INTR    intr/s
09:38:16             1      1,00
09:38:17             1      0,00
09:38:18             1      1,00
09:38:19             1     24,00
09:38:20             1     30,00
09:38:21             1     29,00
09:38:22             1     29,00
09:38:23             1     29,00
09:38:24             1     29,00
09:38:25             1     30,00
09:38:26             1     29,00
09:38:27             1     29,00
09:38:28             1      3,00
^C
Średnia:             1     20,23
```

Monitorowanie przerwań sprzętowych: myszka/gładzik

Działanie: szybkie ruchy myszką

```
root@nscobie ~$ sar -I 12 1
Linux 5.14.16-101.fc33.x86_64 (nscobie)         15.11.2021      _x86_64_        (8 CPU)

09:45:56         INTR    intr/s
09:45:57           12      0,00
09:45:58           12     71,00
09:45:59           12    469,00
09:46:00           12    471,00
09:46:01           12    471,00
09:46:02           12    469,00
09:46:03           12    472,00
09:46:04           12    469,00
09:46:05           12    473,00
09:46:06           12    349,00
09:46:07           12    356,00
09:46:08           12    472,00
09:46:09           12    102,00
09:46:10           12      0,00
^C
Średnia:           12    273,02
```


Monitorowanie przerwań sprzętowych: wlan0

Działanie: `ping -f 192.168.1.1`

```
root@nscobie ~$ sar -I 147 1
Linux 5.14.16-101.fc33.x86_64 (nscobie)         15.11.2021      _x86_64_        (8 CPU)

09:35:02         INTR    intr/s
09:35:03          147      2,00
09:35:04          147     62,00
09:35:05          147    412,00
09:35:06          147    425,00
09:35:07          147    414,00
09:35:08          147    425,00
09:35:09          147    403,00
09:35:10          147    388,00
09:35:11          147    410,00
09:35:12          147    414,00
09:35:13          147    428,00
09:35:14          147    383,00
^C
Średnia:          147    278,47
```


Monitorowanie przerwań sprzętowych: eth0

Działanie: `ping -f 192.168.1.1`

```
root@nscobie ~$ sar -I 149 1
Linux 5.14.16-101.fc33.x86_64 (nscobie)         15.11.2021      _x86_64_        (8 CPU)

09:32:45         INTR    intr/s
09:32:46          149      2,00
09:32:47          149      8,00
09:32:48          149   4327,00
09:32:49          149  14431,00
09:32:50          149  14697,00
09:32:51          149  13461,00
09:32:52          149  13600,00
09:32:53          149   1674,00
09:32:54          149      0,00
09:32:55          149     20,00
09:32:56          149      5,00
09:32:57          149     15,00
^C
Średnia:          149   3663,06
```


Synchroniczne i asynchroniczne wejście-wyjście

System operacyjny zlecając wykonanie operacji wej-wyj musi wiedzieć, kiedy operacja została zakończona

- synchroniczne operacje wej-wyj – po zleceniu wykonania operacji procesor czeka na jej zakończenie
- asynchroniczne operacje wej-wyj – po zleceniu wykonania operacji procesor przetwarza kolejne zadanie do czasu nadejścia przerwania z urządzenia

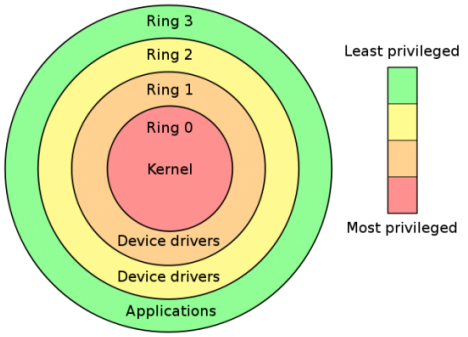
W systemach wieloprogramowych i z podziałem czasu operacje wej-wyj nakłada się na działanie jednostki centralnej (asynchroniczne wej-wyj).

Ochrona sprzętowa

- generowanie przerwania od czasomierza (co kwant czasu)[32]
- ochrona pamięci realizowana przez sprzęt (stronicowanie, dawniej adres bazowy i graniczny)
- ochrona wektora przerwań, procedur wej-wyj
- wszelkie rozkazy wej-wyj są uprzywilejowane (pierścienie uprzywilejowania)

[32] W jądrach >= 2.6 pojawia się możliwość zastosowania tzw. *dynamic ticks*, tzn. że przerwania zegarowe są generowane w razie potrzeby zarówno wtedy, kiedy system jest zajęty, jak i wtedy kiedy pracuje w trybie "idle"(CPU idle state).

Pierścienie uprzywilejowania[33]



[33] http://en.wikipedia.org/wiki/Ring_(computer_security).

Dualny tryb pracy

- system operacyjny musi gwarantować, że niepoprawny program nie będzie mógł zakłócić działania innych programów
- niepoprawnie działający program musi generować przerwanie
- ochronie muszą podlegać wszelkie zasoby dzielone
- sprzęt pozwalający odróżnić dwa tryby pracy: tryb użytkownika oraz tryb monitora (nadzorcy, systemu)
- każde przerwanie powoduje przejście systemu z trybu użytkownika do trybu monitora
- dualny tryb pracy jest uzupełniony listą uprzywilejowanych rozkazów maszynowych, które mogą być wykonywane tylko w trybie monitora

Dualny tryb pracy (cd)

- **tryb nadzorcy (monitora)** – w tym trybie wykonywane są rozkazy uprzywilejowane, np. rozkazy wejścia-wyjścia, zmieniające stan rejestrów zarządzających pamięcią i czasomierzem, rozkaz *halt*, rozkazy włączania/wyłączania przerwań
- **tryb użytkownika** – program użytkownika wykonuje rozkazy uprzywilejowane poprzez wywołanie systemowe (*system calls*).

  Odwołanie do systemu jest traktowane przez sprzęt jak przerwanie programowe. Poprzez wektor przerwań sterowanie przekazywane jest do odpowiedniej procedury obsługi w systemie operacyjnym.

  Rodzaj usługi, na którą zgłasza zapotrzebowanie użytkownik jest określony przez parametr odwołania do systemu.

  System operacyjny wykonuje na życzenie użytkownika operacje zastrzeżone dla nadzorcy.

Przerwania programowe i wywołania systemowe

Z punktu widzenia programisty wywołania systemowe nie istnieją. Programista korzysta jedynie z programowego interfejsu aplikacji (API *Application Programming Interface*) dostarczanego przez system.[34]
Interfejs wywołań systemowych w Linuksie (większości Uniksów) stanowi częściowo biblioteka C.[35]

Program:	wywołanie funkcji printf()
Biblioteka C:	printf() ⟶ write()
Jądro:	wywołanie systemowe write()

Wywołania systemowe są niezbędne, gdyż programy działające w przestrzeni użytkownika nie mogą wykonać kodu jądra. Programy nie mogą bezpośrednio zainicjować wywołania funkcji operacji jądra, ponieważ jądro jest umieszczone i działa w chronionym obszarze pamięci operacyjnej.

[34] API Linuksa jest zgodny ze standardem POSIX (*Portable Operating System Interface*).
[35] Nie jest to rozwiązanie bezproblemowe, zob. https://lwn.net/Articles/771441/.

Przerwania programowe i wywołania systemowe

Przerwanie programowe realizuje w procesorach rodziny x86 instrukcja `int $0x80`. Powoduje ona przełączenie systemu do trybu jądra i rozpoczęcie wykonywania wektora przerwania 128, który wskazuje na adres procedury obsługi wywołań systemowych `system_call()` (zdefiniowana w pliku *entry.S*).

Numer identyfikujący wywołanie systemowe jest umieszczany w rejestrze `eax`. Rejestry (`ebx, ecx, edx, esi, edi`) są (zwykle) używane do przekazywania argumentów wywołania systemowego. Przed użyciem dokonywana jest dokładna weryfikacja przekazanych argumentów.

Przerwania programowe i wywołania systemowe (cd)

Fragmenty pliku `/usr/include/asm/unistd_32.h`

Fedora 22

```
#define __NR_restart_syscall 0
#define __NR_exit 1
#define __NR_fork 2
#define __NR_read 3
#define __NR_write 4
#define __NR_open 5
#define __NR_close 6
#define __NR_waitpid 7
#define __NR_creat 8
#define __NR_link 9
#define __NR_unlink 10
#define __NR_execve 11
#define __NR_chdir 12
#define __NR_time 13
#define __NR_mknod 14
#define __NR_chmod 15
#define __NR_lchown 16
...
#define __NR_kcmp 349
#define __NR_finit_module 350
#define __NR_sched_setattr 351
#define __NR_sched_getattr 352
#define __NR_renameat2 353
#define __NR_seccomp 354
#define __NR_getrandom 355
#define __NR_memfd_create 356
#define __NR_bpf 357
#define __NR_execveat 358
```

Fedora 26

```
...
#define __NR_getrandom 355
#define __NR_memfd_create 356
#define __NR_bpf 357
#define __NR_execveat 358
#define __NR_socket 359
#define __NR_socketpair 360
#define __NR_bind 361
#define __NR_connect 362
#define __NR_listen 363
#define __NR_accept4 364
...
#define __NR_sendto 369
#define __NR_sendmsg 370
#define __NR_recvfrom 371
#define __NR_recvmsg 372
#define __NR_shutdown 373
#define __NR_userfaultfd 374
#define __NR_membarrier 375
#define __NR_mlock2 376
#define __NR_copy_file_range 377
#define __NR_preadv2 378
#define __NR_pwritev2 379
#define __NR_pkey_mprotect 380
#define __NR_pkey_alloc 381
#define __NR_pkey_free 382
#define __NR_statx 383
#define __NR_arch_prctl 384
```


Przerwania programowe i wywołania systemowe

Fragmenty pliku `/usr/include/asm/unistd_64.h`

Fedora 24

```
#define __NR_read 0
#define __NR_write 1
#define __NR_open 2
#define __NR_close 3
#define __NR_stat 4
#define __NR_fstat 5
#define __NR_lstat 6
...
#define __NR_mmap 9
#define __NR_mprotect 10
#define __NR_munmap 11
#define __NR_brk 12
...
#define __NR_access 21
#define __NR_pipe 22
#define __NR_select 23
#define __NR_sched_yield 24
...
#define __NR_clone 56
#define __NR_fork 57
#define __NR_vfork 58
#define __NR_execve 59
#define __NR_exit 60
#define __NR_kill 62
#define __NR_uname 63
...
#define __NR_preadv2 327
#define __NR_pwritev2 328
```

Fedora 33

```
...
#define __NR_pkey_free 331
#define __NR_statx 332
#define __NR_io_pgetevents 333
#define __NR_rseq 334
...
#define __NR_pidfd_send_signal 424
#define __NR_pidfd_send_signal 424
#define __NR_io_uring_setup 425
#define __NR_io_uring_enter 426
...
#define __NR_mount_setattr 442
#define __NR_quotactl_fd 443
#define __NR_landlock_create_ruleset 444
#define __NR_landlock_add_rule 445
#define __NR_landlock_restrict_self 446
#define __NR_memfd_secret 447
```

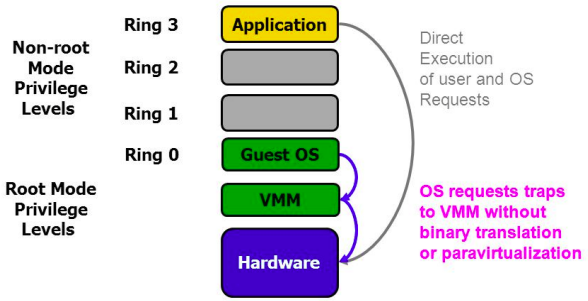
Przerwania programowe: śledzenie

```
$ strace -c cat /etc/hosts > /dev/null
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 35.84    0.000944         944         1           execve
 11.92    0.000314          28        11         6 openat
 11.20    0.000295          29        10           mmap
  6.42    0.000169          24         7           close
  5.58    0.000147          36         4           brk
  5.28    0.000139          23         6           fstat
  4.78    0.000126          31         4           mprotect
  4.71    0.000124          24         5           read
  3.83    0.000101          50         2         1 arch_prctl
  3.49    0.000092          46         2           munmap
  3.26    0.000086          21         4           pread64
  2.01    0.000053          53         1         1 access
  0.91    0.000024          24         1           write
  0.76    0.000020          20         1           fadvise64
------ ----------- ----------- --------- --------- ----------------
100.00    0.002634          44        59         8 total
```


Przerwania programowe: śledzenie

```
$ strace -c firefox
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 26.32    0.364886          45      7963       505 futex
 13.59    0.188327          13     14104     12926 recvmsg
 11.21    0.155370          14     11038           poll
  7.90    0.109478          18      5802           write
  7.60    0.105350          18      5705         1 madvise
  5.28    0.073184        2152        34        16 wait4
  5.19    0.071983          12      5705         3 read
  4.38    0.060772          23      2565           mprotect
  3.09    0.042794          20      2041           mmap
  3.08    0.042744          41      1023           munmap
   ...
  0.00    0.000000           0         1           set_robust_list
  0.00    0.000000           0         1           inotify_init1
  0.00    0.000000           0         1           sched_setattr
  0.00    0.000000           0         1           sched_getattr
------ ----------- ----------- --------- --------- ----------------
100.00    1.386228          20     68379     14862 total
```


Pierścienie uprzywilejowania i wirtualizacja[36]



[36] http://slideplayer.com/slide/5084764/

Procesy

- Proces to program, który jest wykonywany.
- Procesem jest wykonywany program użytkownika, zadanie systemowe (spooling, przydział pamięci, itp.).
- Program jest bierny, jest zbiorem bitów przechowywanych na dysku. Program NIE jest procesem.
- Proces JEST aktywny, dla procesu licznik rozkazów wskazuje następną instrukcję do wykonania. Wykonanie procesu musi przebiegać w sposób sekwencyjny.

Obserwacja procesów: pstree -ch, top, htop

Procesy: zadania systemu operacyjnego

System operacyjny odpowiada za wykonywanie następujących czynności:

- tworzenie i usuwanie procesów
- wstrzymywanie i wznawianie procesów
- dostarczanie mechanizmów komunikacji procesów
- dostarczanie mechanizmów obsługi blokad/rygli

Procesy: rodzaje

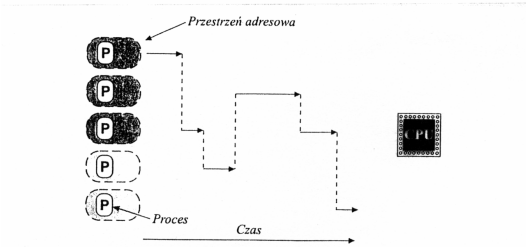
Proces stanowi jednostkę pracy w systemie.

System składa się ze zbioru procesów:

- procesy systemu operacyjnego (wykonują kod systemu)
- procesy użytkowników (wykonują kod programów użytkowników)

Współbieżność (pseudoparalelizm) – w systemie (jednoprocesorowym) z podziałem czasu w każdej chwili wykonuje się tylko jeden proces, ale z uwagi na przełączanie kontekstu powstaje wrażenia równoczesnej pracy wielu procesów.

Współbieżność a przetwarzanie równoległe w systemie wieloprocesorowym.



Rysunek 3-1. Tradycyjny system UNIX — jeden procesor z jednowątkowym procesem

Procesy: zarządzanie

Blok kontrolny procesu:

- stan procesu
- numer procesu
- licznik rozkazów, stosu
- rejestry
- ograniczenia pamięci
- wykaz otwartych plików
- informacja o planowaniu przydziału procesora
- informacja o wykorzystanych zasobach (rozliczanie)

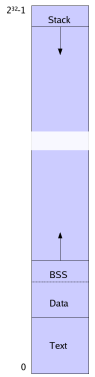
Procesy: zarządzanie w systemie GNU/Linux

Zadanie (*task*) w systemie Linux może być traktowane jako równoważne typowemu jednowątkowemu procesowi w systemie UNIX. Struktura danych opisująca zadanie zawiera atrybuty:

- informacje o zadaniu (*task info*): jednoznaczny identyfikator zadania (ID), informacje o związkach rodzic/dziecko pomiędzy zadaniami, identyfikator grupy zadań (TGID), który pozwala identyfikować zadania związane z jednym procesem
- przestrzeń adresową (*address space*): definiuje przestrzeń wirtualną zadania
- informację o przydziale procesora (*scheduling info*): status zadania, politykę przydziału procesora, parametry tej polityki (priorytet dynamiczny), flaga *need_resched*
- informacja o programie wykonywalnym (*executable info*): identyfikuje plik, który jest obrazem wykonywanego zadania

- informacja o sygnałach (*signal info*): zawiera dane związane z sygnałami i ich obsługą (*signal handler table, pending signal mask, signal queue*)
- dane uwierzytelniające (*credentials*): dane określające prawa i przywileje zadania (UID, GID, maska określająca prawa dostępu do urządzeń)
- informacje księgowe (*accounting info*): czas utworzenia zadania, czas zużyty w trybie jądra i użytkownika, liczba błędów stron, itp.
- ograniczenia zasobów (*resource limits*): parametry określające maksymalną wielkość pliku *core*, czas wykonywania się zadania, wykorzystywaną pamięć, liczbę otwartych plików, itp.
- informacja o systemie plików (*filesystem info*): domyślna maska określająca prawa dostępu przy tworzeniu plików (*umask*), bieżący katalog
- tablica otwartych plików (*open file table*): tablica plików otwartych przez zadanie

Procesy: przestrzeń adresowa



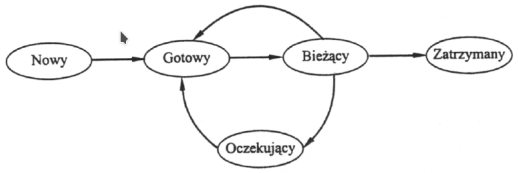
Z procesem związana jest określona wirtualna przestrzeń adresowa, segment programu, danych (zainicjowanych i niezainicjowanych), sterty, stosu.

```
# size /bin/ps
   text  data     bss     dec     hex filename
  71928   769  131967  204664   31f78 /bin/ps
```

```
# pmap PID
```


Procesy: stany

- gotowy – proces czeka na przydział procesora
- bieżący (aktywny) – są wykonywane instrukcje
- oczekujący – proces czeka na wystąpienie jakiegoś zdarzenia (np. zakończenia operacji wejścia-wyjścia)



Komenda ps (*process status*) pozwala obserwować i określać stan procesów w systemie.

```
# ps -elf
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root         1     0  0  80   0 -  1700 epoll_ Nov18 ?        00:00:02 /sbin/init
1 S root         2     0  0  80   0 -     0 kthrea Nov18 ?        00:00:00 [kthread]
1 S root         3     2  0  80   0 -     0 run_ks Nov18 ?        00:00:01 [ksoftirqd/0]
...
0 S jkob      1679     1  0  80   0 -   943 poll_s Nov18 ?        00:00:00 /usr/libexec/gam_server
1 S jkob      1686     1  0  80   0 - 21924 poll_s Nov18 ?        00:00:01 /usr/bin/kglobalaccel
1 S jkob      1688     1  0  80   0 - 22676 poll_s Nov18 ?        00:00:01 /usr/bin/kwalletd
1 S jkob      1692     1  0  80   0 - 21770 poll_s Nov18 ?        00:00:01 /usr/bin/kactivitymanagerd
4 S root      1694     1  0  80   0 -  6969 poll_s Nov18 ?        00:00:16 /usr/libexec/upowerd
0 S jkob      1713  1470  0  80   0 -   542 unix_s Nov18 ?        00:00:00 kwrapper4 ksmserver
0 S jkob      1714  1674  0  80   0 - 37338 poll_s Nov18 ?        00:00:01 ksmserver
4 S root      1736     1  0  80   0 -  6077 poll_s Nov18 ?        00:00:13 /usr/libexec/udisks-daemon --no-debug
1 S root      1737  1736  0  80   0 -  1652 poll_s Nov18 ?        00:00:00 udisks-daemon: not polling any devices
...
0 S jkob     31307  1776  0  80   0 -  1395 wait   10:44 pts/49   00:00:00 /bin/bash
4 S root     31317 31307  0  80   0 -  2384 wait   10:44 pts/49   00:00:00 su - root
4 S root     31322 31317  0  80   0 -  1888 n_tty_ 10:44 pts/49   00:00:00 -bash
0 S jkob     32527  1776  0  80   0 -  1395 wait   11:30 pts/47   00:00:00 /bin/bash
4 S root     32529 32527  0  80   0 -  2384 wait   11:30 pts/47   00:00:00 su - root
4 S root     32534 32529  0  80   0 -  1888 n_tty_ 11:30 pts/47   00:00:00 -bash
0 S jkob     32738  1776  0  80   0 -  1395 wait   11:31 pts/52   00:00:00 /bin/bash
4 S root     32740 32738  0  80   0 -  2384 wait   11:31 pts/52   00:00:00 su - root
4 S root     32745 32740  0  80   0 -  1930 n_tty_ 11:31 pts/52   00:00:00 -bash
```


Komenda ps pozwala obserwować i określać stan procesów w systemie (najnowsze jądro).

```
# ps -elf
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root         1     0  0  80   0 - 55324 ep_pol Nov10 ?        00:00:15 /usr/lib/systemd/systemd
1 S root         2     0  0  80   0 -     0 kthrea Nov10 ?        00:00:00 [kthreadd]
1 S root         4     2  0  60 -20 -     0 worker Nov10 ?        00:00:00 [kworker/0:0H]
1 S root         6     2  0  60 -20 -     0 rescue Nov10 ?        00:00:00 [mm_percpu_wq]
...
4 S avahi      820     1  0  80   0 - 12035 poll_s Nov10 ?        00:00:12 avahi-daemon: running [sc
4 S root       822     1  0  80   0 - 98305 poll_s Nov10 ?        00:00:03 /usr/libexec/accounts-dae
4 S rtkit      823     1  0  81   1 - 45995 poll_s Nov10 ?        00:00:00 /usr/libexec/rtkit-daemon
4 S root       824     1  0  80   0 -  6329 hrtime Nov10 ?        00:00:00 /usr/sbin/smartd -n -q ne
...
0 S jkob      1987  1842  0  80   0 - 30643 poll_s Nov10 pts/2    00:00:00 /bin/bash
...
4 R root     32476 25560  0  80   0 - 36787 -      15:42 pts/22   00:00:00 ps -elf
```



```
# ps auxfw
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         2  0.0  0.0      0     0 ?        S<   Oct02   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   Oct02   0:00  \_ [migration/0]
...
root       494  0.0  0.0      0     0 ?        S<   Oct02   0:00  \_ [ksnapd]
root       502  0.0  0.0      0     0 ?        S<   Oct02   0:16  \_ [kjournald]
root     27749  0.0  0.0      0     0 ?        S<   22:15   0:00  \_ [ipw2200/0]
root         1  0.0  0.0   2136   640 ?        Ss   Oct02   0:04 init [3]
root       567  0.0  0.0   2988  1352 ?        S<s  Oct02   0:01 /sbin/udevd -d
root      2143  0.0  0.0  12204   660 ?        S<sl Oct02   0:00 auditd
root      2145  0.0  0.0  10988   668 ?        S<s  Oct02   0:00  \_ /sbin/audispd
root      2177  0.0  0.0   1800   636 ?        Ss   Oct02   0:01 syslogd -m 0
root      2180  0.0  0.0   1740   400 ?        Ss   Oct02   0:00 klogd -x
...
jkob     11469  0.3  1.9 162520 40624 pts/10   Sl   Oct09   9:02 amarokapp
jkob     11481  0.0  0.1   4096  2384 pts/10   S    Oct09   0:00  \_ ruby /usr/share/apps/amarok/scripts/score_default/
...
jkob     22279  0.9  5.0 185664 104192 ?       Sl   15:57   4:13  \_ /usr/lib/opera/9.50/opera -style plastik
jkob     22326  2.4  1.3  97144 27520 ?        RNl  15:57  11:05  |   \_ /usr/lib/opera/9.50/operapluginwrapper 119 124
jkob     22327  0.0  0.0   2740   448 ?        S    15:57   0:00  |   \_ /usr/lib/opera/9.50/operapluginclean
...
jkob     28592  0.0  0.0   4896  1104 pts/14   S+   23:29   0:00  |       \_ /bin/sh /usr/lib/openoffice.org/program/s
jkob     28604  8.8  2.4 185340 50048 pts/14   Sl+  23:29   0:02  |           \_ /usr/lib/openoffice.org/program/soffi
```


Procesy: tworzenie (`fork` i `exec`)[37]

Realizując funkcję systemową `fork()` jądro wykonuje następujące operacje:

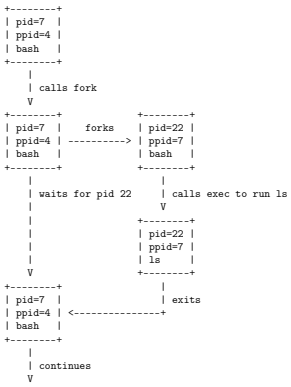
- przydziela nowemu procesowi pozycję w tablicy procesów
- przydziela procesowi potomnemu unikatowy identyfikator
- tworzy logiczną kopię procesu macierzystego (ew. zapewniając współdzielenie segmentów instrukcji, itp.); *copy-on-write*
- zwiększa plikom związanym z tym procesem liczniki w tablicy plików i i-węzłów
- przekazuje identyfikator potomka procesowi macierzystemu i wartość zero procesowi potomnemu

Realizując `exec()`[38] jądro wywołuje inny program, umieszczając w obszarze pamięci procesu kopię pliku wykonywalnego. Zawartość kontekstu poziomu użytkownika staje się niedostępna, z wyjątkiem parametrów `exec()`, które jądro kopiuje ze starej do nowej przestrzeni adresowej.

[37] A. Brouwer, *The Linux kernel*; zob. także fork+exec.sh, fork[1-3].c

[38] Jądro realizuje wywołanie systemowe `execve()`, ale dostępnych jest kilka interfejsów tego wywołania; zob. `man 3 exec`.

Procesy: tworzenie[39]



[39] Exec function and its family

Stany procesów wg *Linux kernel*

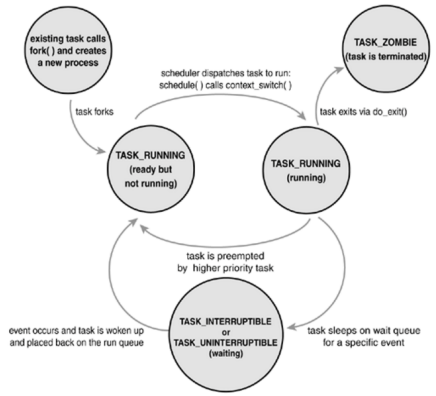
- TASK_RUNNING – proces albo się wykonuje, albo czeka na wykonanie w kolejce procesów gotowych.
- TASK_INTERRUPTIBLE – proces jest wstrzymany do czasu zajścia określonego zdarzenia. Proces może zostać zbudzony przez przerwania sprzętowe, zwolnienie jakiegoś zasobu systemowego, na który proces czeka, albo dostarczenie sygnału.
- TASK_UNINTERRUPTIBLE – stan typu TASK_INTERRUPTABLE, ale z określonym czasem wybudzenia. Wybudzenie może nastąpić po uzyskaniu zasobu, na który proces czekał. Dostarczenie sygnału do uśpionego procesu nie zmienia jego stanu. Tożsamy z TASK_KILLABLE[40].
- TASK_ZOMBIE – wykonanie procesu zostało przerwane, ale proces rodzica nie użył jeszcze wywołania systemowego wait() lub waitpid(), które zwraca informację o przerwanym procesie.
- TASK_STOPPED – wykonanie procesu zostało zatrzymane (wskutek odebrania sygnału SIGSTOP, SIGTSTP, SIGTTIN, SIGTTOU.

[40] TASK_KILLABLE/, Linux kernel: include/linux/sched.h

Stany procesów wg `man ps`

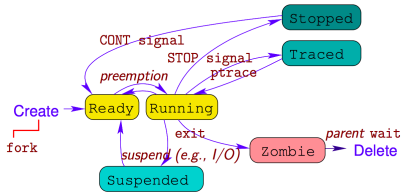
- R (running or runnable) – wykonujący się lub zdolny do pracy (oczekujący w kolejce)
- S (interruptible sleep) – śpiący (blocked)
- D (uninterruptable sleep) – nieprzerywalny sen (zwykle I/O)
- T – zatrzymany przez sygnał sterujący
- t – zatrzymany przez program śledzący (debugger)
- Z (defunct) "zombie" – zakończony, ale nie zebrany przez proces rodzica
- < (high priority process) – proces o wysokim priorytecie
- N (low priority process) – proces o niskim priorytecie
- L (pages locked into memory) – proces ze stronami uwięzionymi w pamięci
- s – leader sesji
- l – proces wielowątkowy
- + – proces pierwszego planu

Stany procesów[41]



[41] http://learnlinuxconcepts.blogspot.com/2014/03/process-management.html

Stany procesów[42]



[42] https://www.enseignement.polytechnique.fr/informatique/INF422/INF422_6.pdf

Wątki

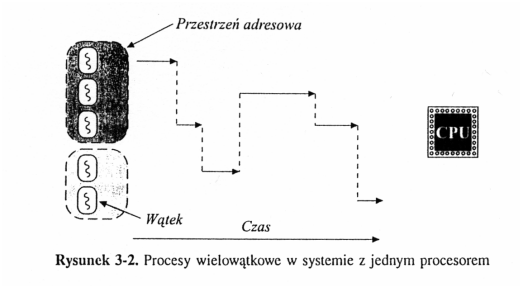
Wątek – (lekki) proces działający w tej samej wirtualnej przestrzeni adresowej, co tworzący go (ciężki) proces. Stan wątku jest zdefiniowany przez małą, odrębną ilość danych (własny stan rejestrów i stos).

Grupa równoprawnych wątków

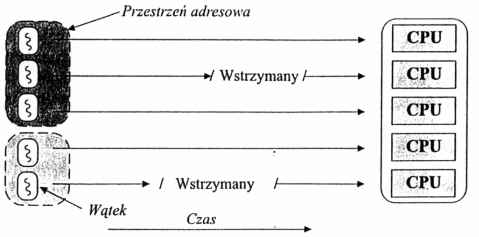
- dzieli kod
- przestrzeń adresową
- otwarte pliki
- zasoby systemu
- należy do tego samego użytkownika

Wątki ze sobą współpracują, a nie współzawodniczą (tak jak procesy).

Wątek, to podstawowa jednostka wykorzystania procesora.



Rysunek 3-2. Procesy wielowątkowe w systemie z jednym procesorem



Rysunek 3-3. Procesy wielowątkowe w wieloprocesorze

Wątki

Zalety:

- przełączanie procesora między wątkami jest łatwiejsze (szybsze) niż między zwykłymi (ciężkimi) procesami
- lepsze wykorzystanie zasobów systemu komputerowego
- lepsza realizacja przetwarzania współbieżnego na maszynach o pamięci współdzielonej (SMP)
- lepsze wykorzystanie procesora poprzez zastosowanie SMT[43]

Rodzaje wątków:

- wątki jądra (*kernel threads*)
- wątki użytkownika (*user threads*), p-wątki (*p-threads*), tj, wątki wg normy POSIX
- procesy lekkie (*lightweight processes*)

[43] *Simultaneous MultiThreading*, zob. J.Corbet, *Core scheduling*

Wątki jądra

Jądro nie jest procesem, ale zarządcą procesów.

Oprócz procesów użytkownika istnieje kilka uprzywilejowanych procesów zwanych *wątkami jądra*, które

- działają w trybie jądra (w przestrzeni adresowej jądra)
- nie komunikują się z użytkownikami (nie trzeba terminali)
- tworzone są w chwili startu systemu i działają do czasu wyłączenia systemu

Wątki użytkownika

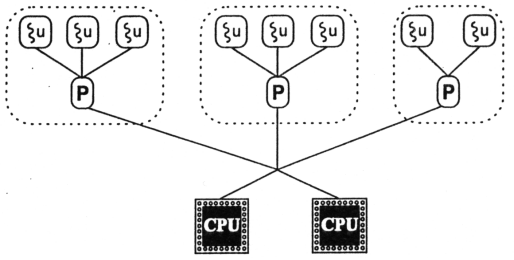
Biblioteka *pthreads* (zgodna ze standardem POSIX) udostępnia abstrakcję wątków całkowicie na poziomie użytkownika. Umożliwia ona tworzenie, usuwanie, synchronizowanie, szeregowanie oraz zarządzanie wątkami bez udziału jądra.

Kontekst wątku z poziomu użytkownika można zapamiętać i odtworzyć bez udziału jądra. Każdy wątek użytkownika ma własny stos, przestrzeń do zapisania kontekstu rejestrów, inne informacje. Jądro jest nadal odpowiedzialne za przełączanie procesów (jądro nie wie o istnieniu wątków użytkownika); jądro wywłaszczając proces użytkownika wywłaszcza związane z nim wątki.

Wątki użytkownika są wydajne, nie zużywają zasobów jądra (jeśli nie są związane z procesem lekkim).

Problemy: biblioteka szereguje wątki, jądro procesy; synchronizacja i ochrona; wątki zwiększają poziom współbieżności, ale nie równoległości.

Procesy i wątki



Procesy lekkie[44]

Proces lekki jest wspieranym przez jądro wątkiem z przestrzeni użytkownika. System udostępniając procesy lekkie musi także udostępniać wątki jądra. W każdym procesie może być kilka procesów lekkich, z których każdy jest wspierany przez osobny wątek jądra.

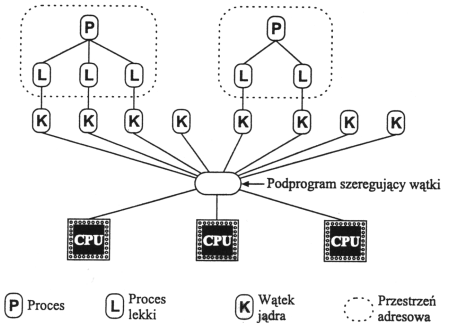
Procesy lekkie są niezależnie szeregowane, współdzielą przestrzeń adresową, mogą wywoływać funkcje systemowe, które powodują wstrzymanie w oczekiwaniu na wejście-wyjście lub zasób. Mogą się wykonywać na różnych procesorach.

Operowanie na procesach lekkich jest kosztowne, gdyż wymaga użycia wywołań systemowych (przełączeń trybu). Trzeba zapewnić synchronizację w dostępie do współdzielonych danych.

Zadanie (*task*) z poziomu trybu użytkownika to wątek, a z poziomu jądra to proces lekki.

[44] https://www.thegeekstuff.com/2013/11/linux-process-and-threads/

Procesy, procesy lekkie i wątki[45]



[45] Implementing threads

Procesy i wątki: porównanie

- **Null fork:** czas mierzony od utworzenia poprzez jego szeregowanie, wykonywanie, aż do zakończenia procesu (wątku), który wykonuje pustą procedurę (miara obciążenia spowodowanego przez tworzenie procesu (wątku))
- **Signal-Wait:** czas potrzebny, aby proces (wątek) przesłał sygnał oczekiwania innemu procesowi (wątkowi) i otrzymał odpowiedź zawierającą warunek (miara obciążenia spowodowanego wzajemną synchronizacją)

Czas oczekiwania (μs) na wykonanie operacji przez wątki użytkownika (UT), wątki jądra (KT) i procesy (P).[46]

	UT	KT	P
Null fork	34	948	11300
Signal-wait	37	441	1840

[46] VAX, system uniksopodobny

Procesy i wątki: obserwowanie

```
# ps -eo user,stat,pid,ppid,tgid,lwp,nlwp,cmd
USER     STAT   PID  PPID  TGID   LWP NLWP CMD
...
named    Ssl  19053     1 19053 19053    5 /usr/sbin/named -u named -c /etc/named.conf
...

# ps -eLfj
UID        PID  PPID  PGID   SID   LWP  C NLWP STIME TTY          TIME CMD
...
named    19053     1 19053 19053 19053  0    5 Nov15 ?        00:00:00 /usr/sbin/named -u named -c /etc/named.conf
named    19053     1 19053 19053 19054  0    5 Nov15 ?        00:55:08 /usr/sbin/named -u named -c /etc/named.conf
named    19053     1 19053 19053 19055  0    5 Nov15 ?        00:55:03 /usr/sbin/named -u named -c /etc/named.conf
named    19053     1 19053 19053 19056  0    5 Nov15 ?        00:03:06 /usr/sbin/named -u named -c /etc/named.conf
named    19053     1 19053 19053 19057  0    5 Nov15 ?        00:21:58 /usr/sbin/named -u named -c /etc/named.conf
...
```

Procesy i wątki: obserwowanie

```
# ps -eo user,stat,pid,ppid,tgid,lwp,nlwp,cmd
USER     STAT   PID  PPID  TGID   LWP NLWP CMD
...
apache   S     8912 22426  8912  8912    1 /usr/sbin/httpd -DFOREGROUND
apache   S     8913 22426  8913  8913    1 /usr/sbin/httpd -DFOREGROUND
apache   S     8914 22426  8914  8914    1 /usr/sbin/httpd -DFOREGROUND
apache   S     8915 22426  8915  8915    1 /usr/sbin/httpd -DFOREGROUND
apache   S     8916 22426  8916  8916    1 /usr/sbin/httpd -DFOREGROUND
root     Ssl  22426     1 22426 22426    2 /usr/sbin/httpd -DFOREGROUND
apache   S    32599 22426 32599 32599    1 /usr/sbin/httpd -DFOREGROUND
...

# ps -efLj
UID        PID  PPID  PGID   SID   LWP  C NLWP STIME TTY          TIME CMD
...
apache    8912 22426 22426 22426  8912  0    1 Nov28 ?        00:00:04 /usr/sbin/httpd -DFOREGROUND
apache    8913 22426 22426 22426  8913  0    1 Nov28 ?        00:00:04 /usr/sbin/httpd -DFOREGROUND
apache    8914 22426 22426 22426  8914  0    1 Nov28 ?        00:00:04 /usr/sbin/httpd -DFOREGROUND
apache    8915 22426 22426 22426  8915  0    1 Nov28 ?        00:00:04 /usr/sbin/httpd -DFOREGROUND
apache    8916 22426 22426 22426  8916  0    1 Nov28 ?        00:00:04 /usr/sbin/httpd -DFOREGROUND
root     22426     1 22426 22426 22426  0    2 Nov16 ?        00:01:28 /usr/sbin/httpd -DFOREGROUND
root     22426     1 22426 22426  8899  0    2 Nov28 ?        00:00:01 /usr/sbin/httpd -DFOREGROUND
apache   32599 22426 22426 22426 32599  0    1 Nov28 ?        00:00:05 /usr/sbin/httpd -DFOREGROUND
...
```


Procesy: tryby pracy

Zmiana trybu pracy zachodzi, gdy:

- proces wywołuje funkcję systemową
- CPU wykonujący proces sygnalizuje *wyjątek*, np. wykonanie nieprawidłowej instrukcji; jądro obsługuje wyjątek na rzecz procesu, który go spowodował
- urządzenie zewnętrzne zgłasza procesorowi *sygnał przerwania*, np. zmiana statusu, zakończenie operacji wej/wyj, itp; urządzenia działają asynchronicznie, więc przerwania nadchodzą w nieprzewidywalnych momentach

Po nadejściu przerwania wykonywany jest wątek jądra. Działa on w trybie jądra, więc odpowiadający mu program musi być uważany za część jądra, chociaż umieszczoną w procesie.

Wywłaszczanie procesu

Kiedy proces wchodzi w stan `TASK_RUNNING` jądro sprawdza, czy jego priorytet nie jest wyższy od priorytetu procesu właśnie zawłaszczającego procesor. Jeśli tak, to wywoływany jest planista wybierający nowy proces do uruchomienia.

Kiedy proces wyczerpie swój kwant czasu, podlega wywłaszczeniu, a planista wybiera kolejny proces do uruchomienia.

Wywłaszczenie procesu użytkownika może nastąpić przy powrocie do przestrzeni użytkownika z wywołania systemowego lub z procedury obsługi przerwania. Wywłaszczenie może zajść także wtedy, kiedy jednostka centralna wykonuje kod w trybie użytkownika.

Procesy: wywłaszczanie

Czy proces może być wywłaszczony, jeśli przebywa w trybie jądra?
Jądra jedno/wielowejściowie.

- Jądra uniksowe są *wielobieżne* (wielowejściowe, *reentrant*): kilka procesów może się wykonywać w trybie jądra w tym samym czasie.
- W systemie jednoprocesorowym tylko jeden proces może działać, inne mogą czekać na CPU lub na zakończenie operacji wej/wyj (będąc zablokowanymi w trybie jądra).
- Wystąpienie przerwania sprzętowego pozwala jądru wielobieżnemu na zatrzymanie procesu, nawet jeśli znajduje się on w trybie jądra. Wpływa to na zwiększenie szybkości obsługi urządzeń zewnętrznych.
- Linux (z jądrem w wersji $\leqslant$ 2.4) jest systemem operacyjnym z wywłaszczaniem procesów, ale bez wywłaszczania jądra. Jądro $\geqslant$2.6 jest już wielowejściowe.

Algorytmy przydziału procesora

Jeśli dwa lub więcej procesów znajdują się w stanie *gotowy do wykonania*, to o kolejności przydziału jednostki centralnej decyduje **planista**.

Planista (*scheduler*) powinien gwarantować:

1. sprawiedliwość przydziału CPU
2. dobrą wydajność CPU
3. mały czas odpowiedzi
4. mały czas przetwarzania (*turnaround time*)
5. dużą przepustowość (*throughput*)

Klasyfikacja procesów I:

- związane z urządzeniami I/O (*I/O-bound*)
- związane z procesorem (*CPU-bound*)

Klasyfikacja procesów II:

- interaktywne
- wsadowe
- czasu rzeczywistego

W systemach Unix/Linux planista zwykle jawnie faworyzuje procesy ograniczone przez operacje wejścia-wyjścia.

Systemy operacyjne z procesami:

- bez wywłaszczania (*nonpreemptive*) – planista jest wywoływany tylko wówczas, kiedy proces dobrowolnie zrezygnuje z CPU (wczesne systemy wsadowe stosowały podejście *run to completion*).
- z wywłaszczaniem (*preemptive*) – system operacyjny odbiera procesowi CPU i wywołuje planistę (przerwania zegarowe!)

  Jeśli wykonuje się zadanie o niskim priorytecie, a jest gotowe do wykonania zadanie o wyższym priorytecie, to zadanie o niskim priorytecie jest wywłaszczane.

Planowanie przydziału procesora:

- rotacyjne, cykliczne (*round-robin scheduling*)
- pierwszy nadszedł, pierwszy obsłużony (FIFO, *first in, first out*)
- priorytetowe (polityka planowania i mechanizm planowania)
- najkrótsze zadanie najpierw (SJF, *shortest job first*)
- dwupoziomowe

Algorytm szeregowania w Linuksie[47]

Od wersji jądra 2.5 (11-2001) był używany planista O(1):

- 140 priorytetów: [0,139]; niższa wartość – wyższy priorytet
- zadania czasu rzeczywistego: [0,99]
- zadania normalne szeregowanie wg priorytetów dynamicznych: [100,139]
- polityki przydziału CPU: `SCHED_FIFO`, `SCHED_RR`, `SCHED_OTHER`[48]

W wersji 2.6.23 (10-2007) pojawił się nowy, całkowiecie sprawiedliwy planista (CFS, *Completely Fair Scheduler*) o złożoności O(log N).

[47] Nikita Ishkov, *A complete guide to Linux*
[48] POSIX 1003.1b definiuje te polityki przydziału procesora.

Priorytety czasu rzeczywistego

Dostępne są dwie strategie szeregowania zadań czasu rzeczywistego:

- zadania klasy `SCHED_FIFO` – gotowe do uruchomienia zadanie tej klasy jest szeregowane przez zadaniami `SCHED_OTHER` i nie otrzymują kwantu czasu (wykonują się do zawieszenia lub oddania procesora)
- zadania klasy `SCHED_RR` – zadanie typu `SCHED_FIFO` z określonym kwantem czasowym (może działać tylko do wyczerpania limitu czasowego)

Priorytety zadań czasu rzeczywistego są z zakresu 0-99 i nie ulegają zmianie w czasie wykonywania zadania:

- najpierw są wykonywane zadania czasu rzeczywistego ściśle w kolejności malejących priorytetów
- jeśli nie ma zadań czasu rzeczywistego, to są wykonywanie pozostałe zadania także w kolejności malejących priorytetów[49]

[49] RHEL/CentOS/Fedora ograniczają maksymalne zużycie CPU dla wątków czasu rzeczywistego do 95%; zob. Real Time Scheduler Throttling

Priorytety dynamiczne

Zasada działania:

- czas procesora jest dzielony na epoki
- w pojedynczej epoce każdy z procesów otrzymuje kwant czasu (niekoniecznie równej długości)
- epoka się kończy, gdy każdy z procesów wykorzysta w całości swój kwant czasu
- nowa epoka rozpoczyna się od przeliczenia priorytetów i wielkości kwantów czasu

Priorytety dynamiczne są z zakresu 100-139 i odpowiadają zakresowi mapowania poziomów uprzejmości od -20 do +19.

Priorytet dynamiczny może być modyfikowany przy użyciu komend `nice/snice`, które wyznaczają wartość priorytetów statycznych: od -20 (najwyższy priorytet) do +19 (najniższy).

Priorytety dynamiczne

- Priorytet procesu klasy `SCHED_OTHER` jest modyfikowany w czasie wykonywania procesu (zadania uzależnione od wejścia-wyjścia mają priorytet podwyższany, a zadania zależne od procesora – obniżany).

  Początkowa wartość priorytetu dynamicznego jest określana przez wartość priorytetu statycznego (*poziomu uprzejmości*) i przyjmuje domyślnie wartość 0. Jest ona zmieniana o ± 5 w zależności od interaktywności zadania (stosunek czasu jaki zadanie spędza w zawieszeniu do czasu aktywności jest miarą uzależnienia zadania od operacji wejścia-wyjścia).
- Kwanty czasu są obliczane na podstawie priorytetów dynamicznych.

  Procesy o wyższym priorytecie otrzymują dłuższy kwant czasu.

  Domyślnie proces otrzymuje kwant 100 ms. Ten kwant może być zwiększony do 800 ms (100) lub zmniejszony do 5 ms (139).

  Nowy proces dziedziczy kwant po swoim rodzicu.

Co oznacza PRI?

proces/wątek	jądro	"top/htop"	"ps -eo pri"	"ps -eo rtprio"	"ps -el"	"nice"
`migration`	99	RT	139	99	-40	-
`kworker`	100	0	39	-	60	-20
`bash`	120	20	19	-	80	0
`khugepaged`	139	39	0	-	99	19

	priorytet wg jądra $\geqslant 100$	priorytet wg jądra < 100
top/htop	`PRI='priorytet wg jądra' - 100`	`PRI=RT`
ps -eo pri	`PRI=139 - 'priorytet wg jądra'`	`PRI='priorytet wg jądra' + 40`
ps -el	`PRI='priorytet wg jądra' - 40`	`PRI='priorytet wg jądra' - 139`

Planista O(1) – podsumowanie

- planista wykonuje wszystkie operacje w czasie O(1); wykorzystywanie heurystyk
- zwykle heurystyki działają dobrze, ale są przypadki, kiedy zawodzą; kod planisty musi zawierać obsługę wyjątków
- heurytyki komplikują działanie planisty i sam kod, więc planista traci na uniwersalności

Czy można opracować algorytm planisty, który byłby pozbawiony powyższych wad? I działał lepiej i bardziej sprawiedliwie traktował procesy?

Planista CFS[50]

Cechy nowego planisty

- klasy zadań objętych wspólną polityką przydziału CPU:
  - wstrzymania (*stop*): wątki `migration/0`, `migration/1`, ...
  - ostateczna (*deadline*): `SCHED_DEADLINE`
  - czasu rzeczywistego (*real time*): `SCHED_RR`, `SCHED_FIFO` (RHEL: *Real-time throttling mechanism*)
  - sprawiedliwa (*fair*): `SCHED_NORMAL`, `SCHED_BATCH`, `SCHED_IDLE`
  - bezczynna (*idle*): wątki `kswap0, kswap1`, ...
- całkowicie sprawiedliwy planista: zmiennej długości kwant czasu jest przydzielany zadaniu, które go najbardziej potrzebuje
- szeregowanie według grup zadań (grupowanie wg UID lub CGROUP[51])

[50] Fairness and interactive performance of O(1) and CFS Linux kernel schedulers, Process Scheduling, CFS Design, Inside the Linux 2.6 Completely Fair Scheduler, Tuning the Task Scheduler, CFS: Completely fair process scheduling in Linux, Completely Fair Scheduler and its tuning, `man sched, man chrt`, Documentation/scheduler/sched-design-CFS.txt

[51] *Coscheduling: simultaneous scheduling in control groups*

Planowanie przydziału procesora musi uwzględniać szereg czynników:

- priorytet zadania (klasa szeregowania)
- lokalność danych (pamięć podręczna i NUMA[52])
- grupy kontrolne (*coscheduling*[53])
- zużycie energii
- ...

[52] *Non-Uniform Memory Access*

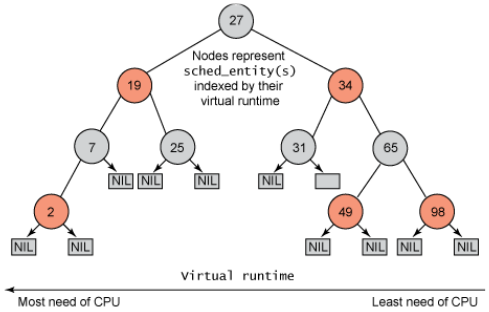
[53] Coscheduling: simultaneous scheduling in control groups

Planista CFS

- każdy z N procesów powinien otrzymać 1/N czasu procesora; wykonujący się proces staje się uprzywilejowany w stosunku do pozostałych; planista wybiera do wykonania proces z najniższym wirtualnym czasem wykonania (VR)[54]
- VR jest rzeczywistym czasem wykonania znormalizowanym do liczby wykonywanych zadań
- zamiast kolejek zadań planista korzysta z drzewa czerwono-czarnego; zadania są porządkowane wg VR i planista wybiera zadanie z najniższą wartością VR
- CFS umożliwia szeregowanie nie tylko pojedynczych zadań, ale grup zadań, np. grupy wątków związanych z jednym ciężkim procesem
- CFS tworzy dla każdego CPU oddzielną strukturę danych (konieczność równoważenia obciążenia)
- nowe zadanie otrzymuje minimalną (aktualną) porcję VR (`min_vruntime`)
- w celu ograniczenia zmian kontekstu minimalna wielkość VR jest ograniczona przez tzw. ziarnistość (*granularity*)

[54] *Virtual Runtime*

Planista CFS[55]



Przykład drzewa czerwono-czarnego wykorzystywanego przez CFS.

[55] Inside the Linux 2.6 Completely Fair Scheduler/

Planista CFS: strojenie

```
# uname -r
3.10.0-1160.45.1.el7.x86_64

# sysctl -a |& grep ^kernel.sched
kernel.sched_autogroup_enabled = 0
kernel.sched_cfs_bandwidth_slice_us = 5000
kernel.sched_child_runs_first = 0
...
kernel.sched_domain.cpu0.domain0.newidle_idx = 0
kernel.sched_domain.cpu0.domain0.wake_idx = 0
...
kernel.sched_latency_ns = 12000000
kernel.sched_migration_cost_ns = 500000
kernel.sched_min_granularity_ns = 10000000
kernel.sched_nr_migrate = 32
kernel.sched_rr_timeslice_ms = 100
kernel.sched_rt_period_us = 1000000
kernel.sched_rt_runtime_us = 950000
kernel.sched_schedstats = 0
kernel.sched_shares_window_ns = 10000000
kernel.sched_time_avg_ms = 1000
kernel.sched_tunable_scaling = 1
kernel.sched_wakeup_granularity_ns = 15000000
```


Planista CFS: strojenie

```
# uname -r
6.0.8-200.fc36.x86_64

# sysctl -a |& grep ^kernel.sched
kernel.sched_autogroup_enabled = 1
kernel.sched_cfs_bandwidth_slice_us = 5000
kernel.sched_child_runs_first = 0
kernel.sched_deadline_period_max_us = 4194304
kernel.sched_deadline_period_min_us = 100
kernel.sched_energy_aware = 1
kernel.sched_rr_timeslice_ms = 100
kernel.sched_rt_period_us = 1000000
kernel.sched_rt_runtime_us = 950000
kernel.sched_schedstats = 0

# ls /sys/kernel/debug/sched
debug domains features idle_min_granularity_ns latency_ns latency_warn_ms
latency_warn_once migration_cost_ns min_granularity_ns nr_migrate
numa_balancing preempt tunable_scaling verbose wakeup_granularity_ns
```

Planista CFS: strojenie

```
# uname -r
5.15.0-53-generic #59-Ubuntu

# sysctl -a |& grep ^kernel.sched
kernel.sched_autogroup_enabled = 1
kernel.sched_cfs_bandwidth_slice_us = 5000
kernel.sched_child_runs_first = 0
kernel.sched_deadline_period_max_us = 4194304
kernel.sched_deadline_period_min_us = 100
kernel.sched_energy_aware = 1
kernel.sched_rr_timeslice_ms = 100
kernel.sched_rt_period_us = 1000000
kernel.sched_rt_runtime_us = 950000
kernel.sched_schedstats = 0
kernel.sched_util_clamp_max = 1024
kernel.sched_util_clamp_min = 1024
kernel.sched_util_clamp_min_rt_default = 1024

# ls /sys/kernel/debug/sched
debug domains features latency_ns latency_warn_ms latency_warn_once
migration_cost_ns min_granularity_ns nr_migrate numa_balancing
tunable_scaling verbose wakeup_granularity_ns
```


Komunikacja międzyprocesowa: sygnały

Sygnały są krótkimi wiadomościami, które można wysyłać do procesu lub grupy procesów. Z każdym sygnałem jest związana jego nazwa i numer (zależne od platformy).

Sygnały:

- zostały wprowadzone w celu ułatwienia komunikacji międzyprocesowej
- są programowymi odpowiednikami przerwań sprzętowych; niektóre przerwania sprzętowe (np. nieprawidłowy adres pamięci) są zamieniane na odpowiednie sygnały
- są dostępne dla procesów w trybie użytkownika
- informują proces o wystąpieniu określonego zdarzenia
- zmuszają proces do wykonania zawartej w kodzie procesu funkcji obsługującej sygnał

Sygnały

Jądro używa sygnałów, żeby powiadomić proces (zadanie) o wystąpieniu błędów lub asynchronicznych zdarzeń.

Przykłady:

- naruszenie ochrony pamięci (SIGSEGV, *segmentation violation signal*)
- przerwanie klawiaturowe Ctrl-C (SIGINT)

Asynchroniczna natura sygnałów powoduje, że proces musi mieć możliwość maskowania (wstrzymywania) wybranych sygnałów; deskryptor procesu (zadania) zawiera maskę.

Sygnały

Linux używa 31 standardowych sygnałów uniksowych i do 32 sygnałów czasu rzeczywistego określonych przez normę POSIX.

Działania podejmowane przez proces po otrzymaniu sygnału:

1. zignorowanie sygnału
2. podejmowanie działań domyślnych
   - przerwanie – proces jest niszczony
   - zrzut – tworzony jest plik `core`[56] i proces jest niszczony
   - ignorowanie – sygnał jest pomijany
   - zatrzymanie – proces jest zatrzymywany (przechodzi w stan TASK_STOPPED)
   - kontynuacja – proces ze stanu TASK_STOPPED przechodzi w stan TASK_RUNNING
3. wyłapanie sygnału i obsłużenie go przez procedury obsługi sygnału

[56] Także `core.$$` lub `/var/lib/systemd/coredump/core.sar.1000...`

Dostępne sygnały: `man 7 signal` (`/usr/include/asm/signal.h`)

1†	**SIGHUP**	zawieszenie procesu lub term kontrolującego
2†	**SIGINT**	przerwanie z klawiatury
3†	SIGQUIT	zamknięcie z klawiatury
4†	SIGILL	nieprawidłowa instrukcja
5	SIGTRAP	pułapka dla programu śledzącego
6†	SIGABRT	=SIGIOT nieprawidłowe zakończenie
7	SIGBUS	błąd szyny
8†	SIGFPE	wyjątek zmiennoprzecinkowy
9†	**SIGKILL**	wymuszone zakończenie procesu
10†	SIGUSR1	do wykorzystania przez proces
11†	**SIGSEGV**	nieprawidłowe odwołanie do pamięci
12†	SIGUSR2	do wykorzystania przez proces
13†	SIGPIPE	zapis do potoku, którego nikt nie czyta
14†	SIGALRM	zegar czasu rzeczywistego
15†	**SIGTERM**	zakończenie procesu

16	SIGSTKFLT	błąd stosu koprocesora
17†	SIGCHLD	proces potomny zakończony/zatrzymany
18†	**SIGCONT**	wznawianie zatrzymanego procesu
19†	**SIGSTOP**	zatrzymanie wykonywania procesu
20†	SIGTSTP	zatrzymanie wykonywania procesu z terminala
21†	SIGTTIN	program w tle żąda wejścia z terminala
22†	SIGTTOU	program w tle żąda wejścia na terminal
23	SIGURG	pilne dane w gnieździe
24	SIGXCPU	przekroczenie limitu czasu procesora
25	SIGXFSZ	przekroczenie limitu wielkości pliku
26	SIGVTALRM	zegar wirtualny
27	SIGPROF	zegar programu profilującego
28	SIGWINCH	zmiana rozmiaru okna
29	SIGIO	SIGPOL, SIGLOST gotowość do operacji I/O
30	SIGPWR	awaria źródła zasilania
31	SIGUNUSE	nie używane

Procesy z poziomu powłoki:

- `pstree --ch | more`
- `ps [-elf]`
- `kill -STOP|stop|19 PID`
- `kill -CONT|cont|18 PID`
- `kill -TERM|term|15 PID`
- `skill -KILL|kill|9 top`
- `pkill -9|15 top`

Sygnał KILL/9 nie trafia do wskazanego procesu, ale do procesu init/systemd!

Synchronizacja i obszary krytyczne

Zasoby niepodzielne: większość urządzeń zewnętrznych, pliki zapisywalne, obszary danych, które ulegają zmianom.

Zasoby podzielne: jednostki centralne, pliki tylko do czytania, obszary pamięci, gdzie znajdują się (dzielone) biblioteki, kody programów, itp.

W systemie wieloprocesowym lub wieloprocesorowym (wielowątkowym) mamy do czynienia z (pseudo)równoczesnymi wątkami wykonania.

- Jak zapewnić prawidłowy dostęp procesów do zasobów niepodzielnych, tj. zasobów, z których może korzystać tylko jeden proces (wątek)?
- Jak unikać zakleszczeń przy dostępie procesów do dwóch lub więcej zasobów równocześnie?

Wyścigi i sekcje krytyczne

Przykład 1:

Biuro A widzi, że jest wolne miejsce X i powiadamia o tym swego klienta	
	Biuro B widzi, że jest wolne miejsce X i powiadamia o tym swego klienta
⋮	
Biuro A rezerwuje miejsce X	⋮
	Biuro B rezerwuje miejsce X

Wyścigi i sekcje krytyczne

Przykład 2: (drukarka jako zasób niepodzielny)

Jak działa system drukowania (*print spooler*)?

- Proces, który chce drukować plik umieszcza nazwę pliku w odpowiednim katalogu (*spooler directory*).
- Inny proces, zwany demonem drukowania (*printer daemon*), regularnie sprawdza zawartość tego katalogu, i – jeśli są pliki do wydrukowania – drukuje je.
- Po wydrukowaniu pliku demon drukowania usuwa plik z katalogu.

Katalog spoolera z nieskończoną liczbę wolnych miejsc:

	⋮
3	
4	/tmp/abc.ps
5	/home/xyz/.tcshrc
6	/etc/hosts
7	
	⋮

- `out` wskazuje następny plik do wydrukowania (np. 4)
- `in` następne wolne miejsce na liście (np. 7)

Jeśli w systemie działa tylko jeden proces A, to drukowanie przebiega wg następującego schematu:

1. proces A umieszcza wartość (dzielonej) zmiennej `in` (7) w lokalnej zmiennej `nwmiejsce` (7)
2. proces A wpisuje do pozycji wskazywanej przez zmienną `nwmiejsce` nazwę pliku do wydrukownia (np. procesA.ps)
3. proces A modyfikuje numer wolnej pozycji: `in=nwmiejsce+1` (8)

Jeśli w systemie dwa procesy A i B próbują skorzystać z drukarki, wówczas

1. proces A umieszcza wartość (dzielonej) zmiennej `in` (7) w lokalnej zmiennej `nwmiejsce` (7)
2. planista wstrzymuje działanie procesu A i wznawia działanie procesu B
3. proces B umieszcza wartość (dzielonej) zmiennej `in` (7) w lokalnej zmiennej `nwmiejsce` (7)
4. proces B wpisuje do pozycji wskazywanej przez zmienną `nwmiejsce` nazwę pliku do wydrukownia (np. procesB.ps)
5. proces B modyfikuje numer wolnej pozycji: `in=nwmiejsce+1` (8).
6. planista wstrzymuje proces B i wznawia działanie procesu A
7. proces A wpisuje do pozycji wskazywanej przez zmienną `nwmiejsce` nazwę pliku do wydrukownia (np. procesA.ps)
8. proces A modyfikuje numer wolnej pozycji: `in=nwmiejsce+1` (8)

Proces B nigdy nie doczeka się wydruku!

Wyścigi i sekcje krytyczne

Przykład 3:

Dwa wątki współużytkują zmienną globalną typu całkowitego n i wykonują kod: `n++`, który oznacza

```
1. pobierz aktualną wartość n i umieść ją w rejestrze
2. dodaj 1 do wartości w rejestrze
3. zapisz nową wartość n do pamięci
```



```
Wątek A                          Wątek B
-------------------------        -------------------------
pobierz n (7)                    ...
zwiększ wartość n (7->8)         ...
zapisz n (8)                     ...
...                              pobierz n (8)
...                              zwiększ wartość n (8->9)
...                              zapisz n (9)
-------------------------------------------------------------
Wątek A                          Wątek B
-------------------------        -------------------------
pobierz n (7)                    ...
...                              pobierz n (7)
zwiększ wartość n (7->8)         ...
...                              zwiększ wartość n (7->8)
zapisz n (8)                     ...
...                              zapisz n (8)
```


Wyścigi i sekcje krytyczne

Wspólnie użytkowane zasoby wymagają ochrony przed współbieżnym dostępem, gdyż współbieżność wątków wykonania może prowadzić do powstanie niespójności w danych.

Sytuacje, w których procesy (wątki) czytają oraz modyfikują pewne dzielone dane i rezultat końcowy zależy od tego, kiedy dokładnie każdy z tych procesów (wątków) będzie je czytał/modyfikował, nazywamy **wyścigiem** (*race condition*), hazardem lub przeplotem operacji.

Część programu, w której następuje czytanie i modyfikowanie dzielonych danych, nazywa się **sekcją krytyczną** (*critical section*).

Jak unikać wyścigów?

Zwiększanie wartości zmiennej globalnej powinno być wykonywane w sposób niepodzielny (atomowy, *atomic operation*). Wzajemne wyłączanie polega na zapewnieniu takich warunków działania systemu, by tylko jeden proces mógł korzystać z zasobów niepodzielnych, tj. tylko jeden proces mógł przebywać w sekcji krytycznej (trzeba zagwarantować wzajemne wykluczanie w odniesieniu do sekcji krytycznych).

1. żadne dwa procesy nie mogą przebywać jednocześnie w swojej sekcji krytycznej
2. nie można czynić żadnych założeń, co do szybkości i liczby CPU
3. żaden proces wykonujący się poza sekcją krytyczną nie może blokować innych procesów
4. żaden proces nie może czekać w nieskończoność na wejście do swojej sekcji krytycznej.

Jak zapewnić wzajemne wykluczanie?

- jądro bez wywłaszczania (*Big Kernel Lock*)
- wyłączanie przerwań
- blokady pętlowe (wirujące)
- semafory
- monitory

Wzajemne wyłączanie z aktywnym czekaniem:

- wyłączanie przerwań (procesy użytkownika i jądra)
- zmienne blokujące (*lock variables*)
- ścisła zmienność (*strict alternation*)
- algorytm Petersona
- instrukcja TSL (*Test and Set Lock*)

Wyłączanie przerwań: proces wyłącza przerwania (także zegarowe), kiedy wchodzi do swojej sekcji krytycznej i włącza je, kiedy ją opuszcza.

Wady:

- proces użytkownika nie powinien mieć możliwości wyłączania przerwań
- w systemach wieloprocesorowych wyłączanie przerwań może dotyczyć tylko jednego procesora

Zalety:

- łatwy sposób modyfikowania struktur jądra w spójny sposób

Zmienne blokujące: podane przykłady ilustrują wady tego podejścia

Ścisła zmienność:

Początkowo `turn = 0`

```
         proces A                          proces B
-------------------------------   -------------------------------
while (TRUE) {                    while (TRUE) {
  while (turn != 0) /* wait */;     while (turn != 1)  /* wait */;
  critical section();               critical section();
  turn = 1;                         turn = 0;
  noncritical_section();            noncritical_section();
}                                 }
```

Co sie dzieje, jeśli oba procesy różnią się szybkością?

Algorytm Petersona

```
#define FALSE 0
#define TRUE  1
#define N     2                     /* number of processes */
int turn;                           /* whose turn is it? */
int interested[N];                  /* all values initially zero (FALSE) */

void enter_region (int process)     /* process: who is entering (0 or 1) */
{
  int other;                        /* number of the other process */
  other = 1 - process;              /* the opposite of process */
  interested[process] = TRUE;       /* show that you are interested */
  turn = process;                   /* set flag */
  while (turn == process && interested[other] == TRUE); /* null statement */
}
void leave_region(int process)      /* who is leaving (0 or 1) */
  interested[process] == FALSE;     /* indicate departure from critical region */
}
```


Jak zapewnić wzajemne wykluczanie?

Instrukcja TSL

```
enter_region:
  tsl register,flag   | copy flag to register and set flag to 1
  cmp register,#0     | was flag zero?
  jnz enter_region    | if it was not zero, lock was set, so loop
  ret                 | return to caller; critical region entered

leave_region:
  mov flag,#0         | store a 0 in flag
  ret                 | return to caller
```

Powyższe rozwiązanie działa tak długo jak procesy „nie oszukują" i w odpowiednich momentach wywołują procedury *enter_region* oraz *leave_region*.

Jak zapewnić wzajemne wykluczanie?

Instrukcja TSL: zalety

- nadają się dla dowolnej liczby procesów (maszyny jedno- i wieloprocesorowe (SMP))
- są proste i łatwe w weryfikacji
- nadają się do kontroli wielu sekcji krytycznych

Instrukcja TSL: wady

- algorytm aktywnego czekania
- możliwość wystąpienia zagłodzeń (*starvation*)
- możliwość wystąpienia zakleszczeń

  Jak uniknąć problemu odwróconego priorytetu?
  Patrz: Report_MarsPathFinder.pdf

Jak zapewnić wzajemne wykluczanie bez aktywnego czekania?

Można uniknąć marnowania czasu jednostki centralnej na jałowe sprawdzanie warunku stosując dwie prymitywne operacje: SLEEP oraz WAKEUP.

- SLEEP – odwołanie do systemu, które powoduje zablokowanie procesu wywołującego tę funkcję do czasu, aż nie nadejdzie sygnał budzenia
- WAKEUP – odwołanie do systemu służące do budzenia określonego procesu

Problem producenta i konsumenta (ograniczonego buforu)

```
#define N     100                    /* number of slots in the buffer */
int count = 0;                       /* number of items in the buffer */

void producer(void)
{
  int item;

  while (TRUE) {                     /* repeat forever */
    produce_item(item);              /* generate next item */
    if (count == N) sleep();         /* if buffer is full go to sleep */
    enter_item(item);                /* put item in buffer */
    count = count + 1;               /* increment number of items in buffer */
    if (count == N) wakeup(consumer); /* was buffer empty? */
  }
}
```


Problem producenta i konsumenta (ograniczonego buforu)

```
void consumer(void)
{
  int item;

  while (TRUE) {                     /* repeat forever */
    if (count == 0) sleep();         /* if buffer is empty go to sleep */
    remove_item(item);               /* take item out of buffer */
    count = count - 1;               /* decrement number of items in buffer */
    if (count == N-1) wakeup(producer); /* was buffer full? */
    consume_item(item);              /* print item */
  }
}
```

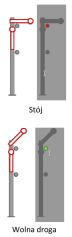
Uwaga! Zmienna count nie jest chroniona, dostęp do niej nie jest niepodzielny.

Semafory

Konieczność zapewnienia niepodzielności wykonywania operacji prymitywnych SLEEP i WAKEUP doprowadziła do powstania semaforów (E.W.Dijkstra, 1965).

Semafor to specjalna zmienna całkowita, na której są wykonywane operacje DOWN i UP:

SLEEP ⟶ DOWN
WAKEUP ⟶ UP



Operacje UP i DOWN muszą być niepodzielne (atomowe), tj. w dowolnej chwili tylko jeden proces/wątek może je wykonywać.

Semafory

Definicje operacji opuszczania (DOWN) i podnoszenia (UP) semafora:

```
void DOWN(semaphore) {
   if (semaphore > 0 ) {
       semaphore = semaphore - 1;
       continue();
   } else {
       sleep();  /* add the process to semaphore queue and block it */
   }
}


void UP(semaphore) {
   semaphore = semaphore + 1;
   wakeup(); /* wakeup all blocked processes and make them runnable */
             /* scheduler selects one to run */
}
```


Problem producenta i konsumenta (rozwiązanie z semaforami)

```
#define N 100                       /* number of slots in the buffer */
typdef int semaphore;               /* semaphores are a special kind of int */
semaphore mutex = 1;                /* controles access to critical section */
semaphore empty = N;                /* count empty buffer slots */
semaphore full  = 0;                /* count full buffer slots */
void producer(void)
{
  int item;
  while (TRUE) {                    /* TRUE is the constant 1 */
    produce_item(&item);            /* generate next item */
    down(&empty);                   /* decrement empty count */
    down(&mutex);                   /* enter critical region */
    enter_item(&item);              /* put new item in buffer */
    up(&mutex);                     /* leave critical region */
    up(&full);                      /* increment count of full slots */
  }
}
```


Problem producenta i konsumenta

```
void consumer(void)
{
  int item;
  while (TRUE) {                    /* infinite loop */
    down(&full);                    /* decrement full count */
    down(&mutex);                   /* enter critical region */
    remove_item(&item);             /* take item from buffer */
    up(&mutex);                     /* leave critical area */
    up(&empty);                     /* increment count of empty slots */
    consume_item(&item)             /* do sth with the item */
  }
}
```


Przeznaczenie semaforów:

- semafor binarny otrzymuje początkową wartość 1 i jest używany przez dwa lub więcej procesów, aby zapewnić że tylko jeden z nich może w danej chwili przebywać w swojej sekcji krytycznej
  - `mutex` jest **semaforem binarnym**
  - `mutex` zapewnienia wzajemne wykluczanie się procesów (*MUTual EXclusion*)
- semafory `empty` oraz `full` służą do **synchronizacji**, zapewniają, że zdarzenia będą zachodziły we właściwej kolejności

Semafory i przerwania

Jak można wykorzystać semafory przy obsłudze przerwań?

- z każdym urządzeniem wej/wyj wiąże się semafor, którego początkowa wartość wynosi zero
- po zainicjowaniu operacji wej/wyj dla danego urządzenia proces wykonuje `DOWN` na semaforze i przechodzi w stan uśpienia
- po nadejściu przerwania proces obsługujący to przerwanie wykonuje `UP` na semaforze
- proces, który zgłosił żądanie wej/wyj staje się gotowy do wykonania

Wady semaforów

- semafor jest wysokopoziomową abstrakcją opartą na niskopoziomowych mechanizmach elementarnych, które zapewniają niepodzielność i dostarczają mechanizmów wstrzymywania
- wstrzymywanie i wznawianie wymaga przełączania kontekstu i zmian w kolejkach modułu szeregowania i kolejk wątków wstrzymanych; operacje na semaforze są **powolne**

Blokady pętlowe, wirujące (*spinlocks*)

- najprostszy elementarny mechanizm blokowania
- główna zaleta: niski koszt
- wątek próbujący pozyskać zasób, aktywnie oczekuje aż do odblokowania zasobu
- wątek aktywnie oczekuje na zasób na jednym procesorze, podczas gdy inny wątek korzysta z tego zasobu na innym procesorze
- wątki nie oddają procesora, jeśli nałożyły blokadę wirującą

blokady wirujące = blokady proste

Monitory

Monitor to programistyczne narzędzie wymuszające wzajemne wykluczanie i koordynację procesów. Realizację struktury monitorów zapewnia język Java (także Concurrent Pascal, Pascal-Plus, Modula).

Monitorem nazywamy moduł programowy składający się z jednej lub kilku procedur, ciągu inicjującego i danych lokalnych o następujących cechach:

1. Zmienne danych lokalnych są dostępne wyłącznie za pośrednictwem procedur monitora (żadna procedura zewnętrzna nie może udostępniać zmiennych lokalnych).
2. Proces uruchamia monitor, wywołując jedną z jego procedur.
3. W tej samej chwili pod kontrolą monitora może działać tylko jeden proces; każdy inny proces, wywołujący w tym samym czasie monitor, musi zostać zawieszony do czasu zwolnienia monitora.

Zakleszczenia (*deadlocks*, *deadly embraces*)

Zbiór procesów jest w stanie zakleszczenia, kiedy każdy z procesów ze zbioru czeka na zdarzenie, które może spowodować inny z procesów.

Zakleszczenie zdarza się, gdy każdy z procesów otrzymał wyłączne prawo do używania jakiegoś zasobu i dodatkowo żąda dostępu do zasobu już zajętego.

Przykłady:

- Proces A otrzymuje dostęp do drukarki, a proces B do dysku, a następnie proces A żąda dostępu do dysku, a proces B do drukarki.
- Proces A otrzymał prawo dostępu do dwóch (lub więcej) rekordów bazy danych, na których operacje mają być przeprowadzane w tym samym czasie przez proces B.

Zakleszczenia

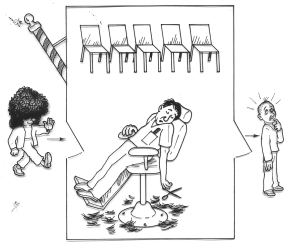
Jak unikać zakleszczeń?

1. Jeśli to tylko możliwe, to należy unikać stosowania blokad (zamiast blokad można stosować operacje atomowe lub używać struktur danych nie wymagających zakładania blokad).
2. Jeśli blokada jest niezbędna, to należy stosować blokadę pojedynczą.
3. Jeśli potrzebnych jest kilka blokad, to należy zdefiniować (małą) lokalną hierarchię blokad lub zastosować blokowanie stochastyczne.

Klasyczne problemy komunikacji międzyprocesowej:

- problem producenta-konsumenta (ograniczonego buforu) – modelowanie procesów synchronizacji między procesami
- problem ucztujących filozofów – modelowania procesów, które współzawodniczą w dostępie do ograniczonych zasobów, np. urządzeń wej/wyj
- problem pisarzy i czytelników – zagadnienie zapewnienia prawidłowego dostępu do bazy danych (w trybie odczytu i zapisu)
- problem śpiącego fryzjera

Problemy ucztujących filozofów i śpiącego fryzjera



Komunikacja międzyprocesowa (IPC)[57]

- potoki i potoki nazwane *(named pipes)*
- kolejkowanie wiadomości *(message queueing)*
- semafory
- współdzielenie pamięci (mapowanie pamięci)
- gniazda (*sockets*)
- zdalne wywołanie procedur *(remote procedure calls (RPC))*

[57] *InterProcess Communication*

Pamięć operacyjna:

- tablica ponumerowanych słów lub bajtów, tzn. każdy element tablicy ma swój adres.
- odgrywa kluczową rolę we współczesnych systemach komputerowych, gdyż jest jedyną pamięcią, którą system operacyjny adresuje bezpośrednio (pamięć swobodnego dostępu, RAM, *Random Access Memory*).
- podczas wykonywania programy oraz ich dane muszą znajdować się w pamięci operacyjnej

Rodzaje systemów zarządzania pamięcią:

- systemy, które cały program w czasie jego wykonania przechowują w pamięci głównej
- systemy, które stale przesuwają procesy pomiędzy pamięcią główną a pamięcią dyskową (wymiatanie i stronicowanie)

Zarządca pamięci jest odpowiedzialny za:[58]

- przydział wolnych obszarów
- odzyskiwanie uwolnionych obszarów
- przenoszenie procesów z pamięci do pamięci pomocniczej
- ochronę obszarów należących do różnych procesów/podsystemów
- sprawiedliwy rozdział pamięci pomiędzy procesy/podsystemy
- mapowanie plików programów wykonywalnych i z danymi na obszary pamięci procesu
- współdzielenie pamięci między procesy
- wirtualizację pamięci

[58] Zwarte omówienie zasadniczych elementów składających się na zarządzanie pamięcią w systemie Linux można znaleźć np. w prezentacji Introduction to Memory Management in Linux

Przenaszalność i ochrona

Wieloprogramowość wymaga przenaszalności (relokowalności) programów oraz ochrony programów różnych użytkowników.

Rozwiązania:

- modyfikowanie słów programu ładowanego do pamięci, które zawierają adresy bezwzględne (OS/360)
- w celu separowania programów użytkowników pamięć dzieli się na kawałki, którym przypisuje się kody ochrony (OS/360)
- zastosowanie rejestrów bazowych i granicznych
- stronicowanie

Systemy z wymianą

W systemach interakcyjnych dostępna pamięć operacyjna jest zwykle mniejsza niż suma pamięci zużywanej przez procesy. Nadmiar procesów musi być przechowywany w pamięci pomocniczej.

Przesuwanie procesów z pamięci głównej na dyskową i z powrotem określa się mianem **wymiany** (*swapping*).

Czas przełączania kontekstu w systemie, w którym stosuje się wymianę jest stosunkowo długi.

Przydział pamięci

Partycje o stałej i zmiennej wielkości (fragmentacja pamięci)

Sposoby kontroli przydziału pamięci operacyjnej:

- mapy bitowe
- połączone listy
- system bliźniaków (*buddy system*)
- alokator płytowy/plastrowy (*slab allocator*); alokator płytowy pobiera grupy wolnych bloków stronicowych wykorzystując algorytm bliźniaków (zob. `slabtop`)

Fragmentacja

- wewnętrzna
- zewnętrzna

Przydział pamięci

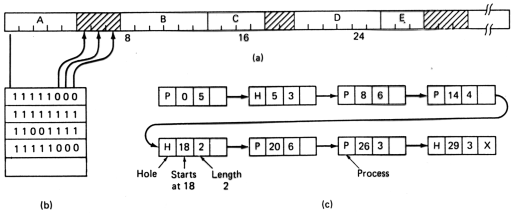
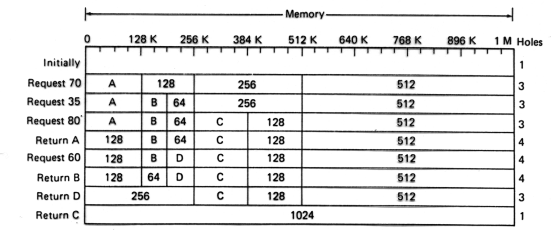


Fig. 3-7. (a) A part of memory with five processes and 3 holes. The tick marks show the memory allocation units. The shaded regions (0 in the bit map) are free. (b) The corresponding bit map. (c) The same information as a linked list.

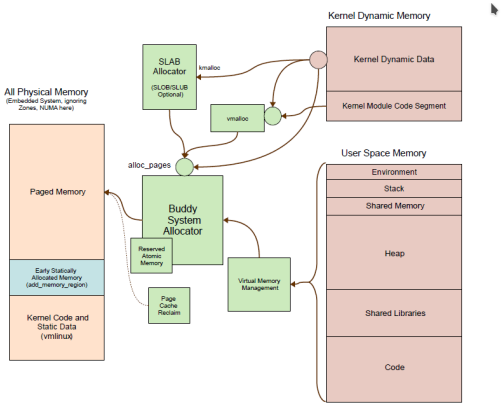
System bliźniaków



Wykonywanie programu większego niż pamięć fizyczna

- nakładkowanie – stosowanie tej metody wymaga dokładnej znajomości struktury programu i starannego zaprojektowania struktury nakładek
- wymiana procesów – jeden lub wiele procesów jest przenoszonych do *pamięci pomocniczej*, aby zrobić miejsce procesowi, któremu planista przydzielił kolejny kwant czasu
- stronicowanie (*paging*) – skuteczny sposób przydziału pamięci procesom z wykorzystaniem nieciągłych obszarów pamięci

Linux: schemat zarządzania pamięcią operacyjną[59]



[59] Xi Wang, Controlling Memory Footprint at All Layers: Linux Kernel, Applications, Libraries, and Toolchain


```
# slabtop
 Active / Total Objects (% used)    : 4484796 / 4739770 (94.6%)
 Active / Total Slabs (% used)      : 117493 / 117493 (100.0%)
 Active / Total Caches (% used)     : 188 / 285 (66.0%)
 Active / Total Size (% used)       : 1057592.76K / 1128933.19K (93.7%)
 Minimum / Average / Maximum Object : 0.01K / 0.24K / 10.06K

  OBJS ACTIVE  USE OBJ SIZE  SLABS OBJ/SLAB CACHE SIZE NAME
837120 836355  99%    0.01K   1635      512      6540K zs_handle
543816 532517  97%    0.07K   9711       56     38844K lsm_inode_cache
530229 510725  96%    0.19K  25249       21    100996K dentry
450723 449121  99%    1.15K  17369       27    555808K ext4_inode_cache
371904 321864  86%    0.10K   9536       39     38144K buffer_head
335420 334657  99%    0.20K  16771       20     67084K vm_area_struct
   ...
 40068  39962  99%    0.59K   1484       27     23744K inode_cache
 39040  37495  96%    0.06K    610       64      2440K kmalloc-64
 35968  34144  94%    0.25K   1124       32      8992K kmalloc-256
 34880  25000  71%    0.50K   1090       32     17440K kmalloc-512
   ...
  4326   3905  90%    0.09K    103       42       412K kmalloc-96
   ...
  2817   2790  99%   10.06K    939        3     30048K task_struct
   ...
  2336   2136  91%    0.12K     73       32       292K pid
   ...
   720    659  91%    1.06K     24       30       768K mm_struct
   ...
   612    540  88%    0.44K     17       36       272K intel_context
   ...
   322    312  96%    0.34K     14       23       112K taskstats
     0      0   0%    1.13K      0       28         0K btrfs_inode
     0      0   0%    0.10K      0       39         0K btrfs_trans_handle
   ...
```


Pamięć wirtualna i fizyczna

Pamięć wirtualna jest abstrakcyjną pamięcią główną w postaci wielkiej, jednorodnej tablicy, która jest logiczna oddzielona od pamięci fizycznej. Procesy nie muszą kłopotać się wielkością dostępnej pamięci fizycznej i jej układem oraz tym, czy jest ona już wykorzystywana przez inne procesy w systemie.

Adresy generowane w trakcie wykonywania programu są adresami w przestrzeni wirtualnej (adresy liniowe). Są one tłumaczone przez specjalny układ elektroniczny (*Memory Managment Unit*, MMU) na adresy w pamięci fizycznej (adresy fizyczne).

Pamięć wirtualna: stronicowanie

- przestrzeń adresów liniowych (wirtualnych) podzielona jest na strony
- pamięć fizyczna podzielona jest na ramki stron (bloki stronicowe)
- strony i ramki stron są zawsze tych samych rozmiarów (od 512 B do 8 KB, w Linuksie 4 KB)
- stronicowanie pozwala na realizację mechanizmu *dzielenia* stron przez dwa lub więcej procesów
- w celu zwiększenia bezpieczeństwa wprowadza się bity ochrony stron (strona zaznaczona np. tylko do odczytu); próba jej zapisu spowoduje wygenerowanie odpowiedniej pułapki i przejęcie kontroli przez system operacyjny
- stronicowanie pozwala na dynamiczne ładowanie potrzebnych fragmentów programów (ładowanie na żądanie)

Pamięć wirtualna i fizyczna: stronicowanie

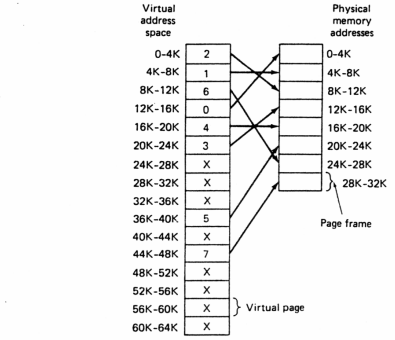


Fig. 3-11. The relation between virtual addresses and physical memory addresses is given by the page table.

Pamięć wirtualna i fizyczna: stronicowanie

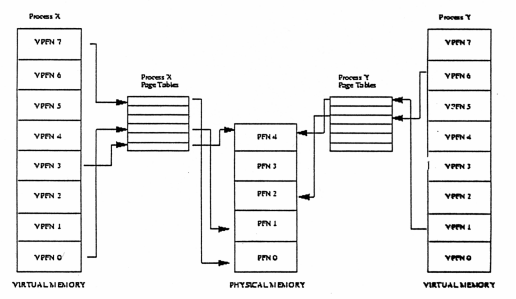


Figure 3.1: Abstract model of Virtual to Physical address mapping

Pamięć wirtualna i fizyczna

Zalety pamięci wirtualnej:

- jednocześnie może być wykonywanych wiele programów
- przestrzenie adresowe poszczególnych procesów są od siebie izolowane (proces może czytać i zapisywać tylko swoje dane)
- możliwe jest wykonywanie programów większych niż dostępna pamięć fizyczna
- procesy mogą wykonywać programy, których kod jest ładowany do pamięci tylko częściowo
- każdy proces może uzyskać dostęp do części dostępnej pamięci fizycznej
- procesy mogą współdzielić w pamięci pojedynczy obraz biblioteki, programu
- programy są relokowalne
- można tworzyć kod niezależny od urządzenia

Struktura pamięci wirtualnej procesu

```
$ pmap.pl -ug top
      87544.1798 G      16K r---- top
      87544.1799 G      72K r-x-- top
      87544.1799 G      24K r---- top
      87544.1799 G       4K r---- top
      87544.1800 G       4K rw--- top
      87544.1800 G     160K rw---   [ anon ]
      87544.1961 G     796K rw---   [ anon ]
     130365.0837 G      12K r---- libnuma.so.1.0.0
     130365.0837 G      24K r-x-- libnuma.so.1.0.0
     ...
     130365.2868 G      20K rw---   [ anon ]
     ...
     130365.2915 G       4K r---- libnss_files-2.30.so
     130365.2915 G     160K rw---   [ anon ]
     130365.2917 G       8K r---- ld-2.30.so
     ...
     130365.2918 G       4K rw---   [ anon ]
     131063.7302 G     132K rw---   [ stack ]
     131063.7304 G      16K r----   [ anon ]
     131063.7305 G       8K r-x--   [ anon ]
17179869183.9902 G       4K r-x--   [ anon ]
```

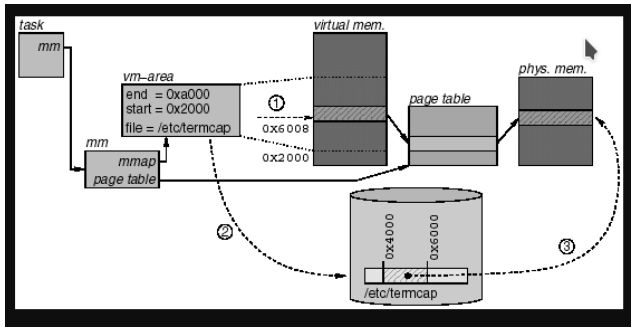


```
# top
Tasks: 607 total,   1 running, 583 sleeping,  22 stopped,   1 zombie
%Cpu(s):  1.4 us,  0.9 sy,  0.0 ni, 97.0 id,  0.0 wa,  0.5 hi,  0.2 si,  0.0 st
MiB Mem :  15069.0 total,    228.8 free,  10786.9 used,   4053.4 buff/cache
MiB Swap:   4096.0 total,      0.0 free,   4096.0 used.   2715.6 avail Mem
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3904 jkob      20   0 1389244 245500 205268 S   3.9   1.6 353:27.38 Xorg
   4307 jkob      20   0 4905708 398004  48032 S   2.6   2.6 192:31.84 plasmashell
   4205 jkob      20   0 3956980 136068  34692 S   2.3   0.9  67:32.76 kwin_x11
 829629 jkob      20   0 1281596  44548  31040 S   1.6   0.3  12:23.86 konsole
  92748 jkob      20   0 7253504 120740  27108 S   0.7   0.8 181:48.44 amarok
 277839 jkob      20   0   16.9g 462504  93604 S   0.7   3.0 517:59.48 brave
    ...
   5084 jkob      20   0 1560068 130868   9876 S   0.3   0.8  64:23.50 seaf-daemon
   5233 jkob      20   0 1569008  49304  27072 S   0.3   0.3   9:44.80 okular
   5546 jkob      20   0  734236  11248   9620 S   0.3   0.1   9:19.77 tracker-miner-r
 277875 jkob      20   0   16.6g  56536  28996 S   0.3   0.4 170:22.55 brave
1079982 root      20   0   13928   8320   7064 S   0.3   0.1   4:34.58 wpa_supplicant
1863224 jkob      20   0   24.6g 166804  68692 S   0.3   1.1  23:59.69 brave
2922530 root      20   0  269000  39232   7612 S   0.3   0.3   0:14.91 sssd_kcm
2980178 root      20   0       0      0      0 I   0.3   0.0   0:00.06 kworker/3:2-events
      1 root      20   0  174732  13344   8332 S   0.0   0.1  11:32.54 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:04.50 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    ...
```


Stronicowanie na żądanie

- Jeśli pamięć fizyczna jest mniejsza od przestrzeni adresowej programu, to nie wszystkie strony programu mogą zostać równocześnie odwzorowane w odpowiednie ramki stron pamięci fizycznej.
- W trakcie wykonywania programu może pojawić się adres do ramki strony, której nie ma w pamięci fizycznej. Wówczas
  - generowany jest sygnał, tzw. błąd strony (*page fault*)
  - w ramach obsługi błędu strony, system przydziela procesowi kolejną ramkę strony i wczytuje do niej odpowiednią stronę programu

Stronicowanie na żądanie[60]



[60] Virtual Memory in the IA-64 Linux Kernel; uwaga na błąd: zamiast 30x6008-0x2000/81923=2 powinno być (0x6008-0x2000)/8192 = 2.0001.

Zarządzanie pamięcią

- brudne strony pamięci (*dirty pages*), tj. takie, które zostały zmodyfikowane, są co pewien czas zapisywane na dysku
- jeśli liczba wolnych stron pamięci spada poniżej zadanego progu, to niewykorzystywane strony czyste są usuwane, a brudne są przenoszone do pamięci pomocniczej
- w razie potrzeby system operacyjny ponownie wczytuje potrzebną stronę z pamięci pomocniczej lub dysku

Zarządzanie pamięcią: `/proc/meminfo`

```
MemTotal:        7843796 kB          NFS_Unstable:          0 kB
MemFree:          334320 kB          Bounce:                0 kB
MemAvailable:    1494104 kB          WritebackTmp:          0 kB
Buffers:          309120 kB          CommitLimit:     9041892 kB
Cached:          1608360 kB          Committed_AS:   17186576 kB
SwapCached:        61052 kB          VmallocTotal:   34359738367
Active:          5235504 kB          VmallocUsed:       36944 kB
Inactive:        1664744 kB          VmallocChunk:          0 kB
Active(anon):    4475700 kB          Percpu:            10688 kB
Inactive(anon):  1170044 kB          HardwareCorrupted:     0 kB
Active(file):     759804 kB          AnonHugePages:         0 kB
Inactive(file):   494700 kB          ShmemHugePages:        0 kB
Unevictable:       26772 kB          ShmemPmdMapped:        0 kB
Mlocked:              16 kB          FileHugePages:         0 kB
SwapTotal:       5119996 kB          FilePmdMapped:         0 kB
SwapFree:        1082488 kB          CmaTotal:              0 kB
Dirty:               220 kB          CmaFree:               0 kB
Writeback:             0 kB          HugePages_Total:       0
AnonPages:       4956496 kB          HugePages_Free:        0
Mapped:           890096 kB          HugePages_Rsvd:        0
Shmem:            662976 kB          HugePages_Surp:        0
KReclaimable:     213092 kB          Hugepagesize:       2048 kB
Slab:             411620 kB          Hugetlb:               0 kB
SReclaimable:     213092 kB          DirectMap4k:     1010552 kB
SUnreclaim:       198528 kB          DirectMap2M:     7073792 kB
KernelStack:       18432 kB          DirectMap1G:           0 kB
PageTables:        79364 kB
```


Zarządzanie pamięcią: `sysctl`[61]

Jądro systemu Linux dostarcza szereg parametrów do strojenia podsytemu zarządzania stronami:

```
vm.dirty_background_bytes = 0
vm.dirty_background_ratio = 10
vm.dirty_bytes = 0
vm.dirty_expire_centisecs = 1500
vm.dirty_ratio = 20
vm.dirty_writeback_centisecs = 1500
vm.dirtytime_expire_seconds = 43200
vm.overcommit_kbytes = 0
vm.overcommit_memory = 0
vm.overcommit_ratio = 50
vm.swappiness = 20
vm.watermark_boost_factor = 15000
vm.watermark_scale_factor = 10
vm.numa_zonelist_order = Node
vm.zone_reclaim_mode = 0
```

[61] https://www.kernel.org/doc/Documentation/sysctl/vm.txt, Tuning memory, Introducing lazytime

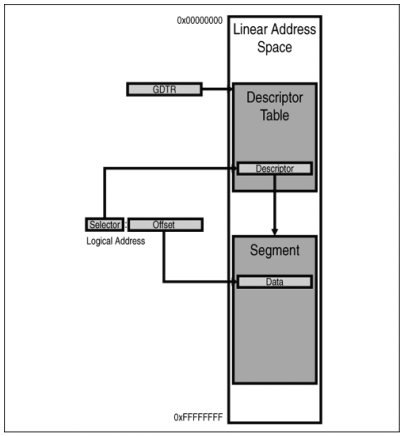
Adresowanie pamięci

W przypadku procesorów Intela mamy trzy rodzaje adresów:

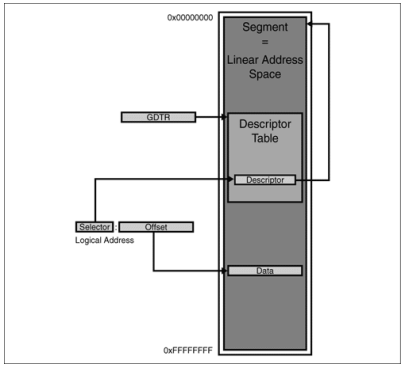
- adres logiczny używany w instrukcjach języka maszynowego do określenia adresów argumentów lub instrukcji; każdy adres składa się z 16-bitowego segmentu i 32-bitowego przesunięcia (offsetu), który oznacza odległość od początku segmentu do aktualnego adresu
- adres liniowy – pojedyncza 32-bitowa liczba całkowita bez znaku, która pozwala adresować do 4 GB (4 294 967 296) komórek pamięci
- adres sprzętowy używany do adresowania komórek pamięci znajdujących się w układach pamięci (32-bitowa liczba całkowita bez znaku)

Sprzętowy obwód o nazwie *jednostka segmentacji* tłumaczy adres logiczny na adres liniowy. Inny obwód sprzętowy, zwany *jednostką stronicowania*, tłumaczy adres liniowy na fizyczny.

Segmentacja w procesorach Intela



Segmentacja w procesorach Intela



Segmentacja w Linuksie

Segmentacja została dołączona do procesorów Intela w celu zachęcenia programistów do dzielenie swoich aplikacji na logicznie połączone jednostki, takie jak podprogramy i globalne i lokalne obszary danych.

Linux używa segmentacji w ograniczonym zakresie, gdyż stosowanie segmentacji i stronicowania jest nadmiarowe:

- zarządzanie pamięcią jest prostsze, gdy wszystkie procesy używają tych samych wartości rejestrów segmentacji, czyli współdzielą ten sam zestaw adresów liniowych
- z założenia Linux ma być przenośnym systemem operacyjnym, a np. część procesorów RISC ma bardzo ograniczone możliwości segmentacji

W architekturze x86_64 segmentacja nie jest używana.

Stronicowanie w procesorach Intela

Poczynając od i80386 jednostki stronicowania obsługują strony o wielkości 4 KB.

32-bitowy adres jest dzielony na trzy pola:

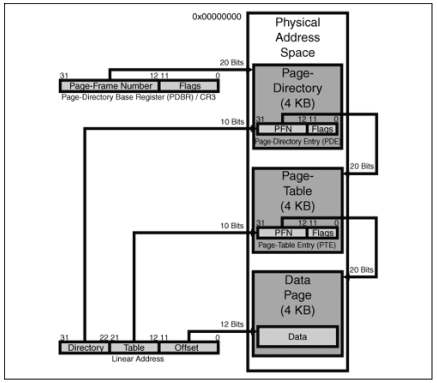
- *Directory* (katalog) najbardziej znaczące 10 bitów
- *Table* (tablica) środkowe 10 bitów
- *Offset* (przesunięcie) najmniej znaczące 12 bitów

Stronicowanie w procesorach Intela

Tłumaczenie adresów liniowych jest oparte na dwóch tablicach: katalogu stron i tablicy stron:

- adres fizyczny aktualnego katalogu stron jest przechowywany w rejestrze cr3 procesora
- pole *directory* w adresie liniowym określa pozycję w katalogu stron, która wskazuje na odpowiednią tablicę stron
- pole *table* w adresie liniowym określa pozycję w tablicy stron, która zawiera adres bloku stronicowego zawierającego daną stronę
- pole *offset* określa pozycję względem początku bloku stronicowego
- 1024 pozycje katalogu stron i 1024 pozycje tablicy stron pozwalają zaadresować $1024 \times 1024 \times 4096 = 2^{32}$ bajtów

Stronicowanie w procesorach Intela



Stronicowanie w Linuksie

Linux jako standardową jednostkę alokacji pamięci przyjmuje bloki stronicowe 4 KB.

Upraszcza to działanie systemu:

- układ stronicowania automatycznie sprawdza, czy zaadresowana strona znajduje się w pewnym bloku stronicowym
- każdy blok stronicowy może być chroniony na poziomie sprzętu za pomocą flagi w pozycji tablicy stron, która na niego wskazuje
- wielkość 4 KB jest wielokrotnością rozmiaru bloku większości dysków, dzięki czemu transfer danych między pamięcią główną a dyskami jest efektywniejszy

Stronicowanie w Linuksie

Linux używa stronicowania trzypoziomowego, co umożliwia działanie architekturom 64-bitowym.

Linux wykorzystuje trzy tablice stronicowania:

- globalny katalog stron (*Global Directory Table*)
- pośredni katalog stron (*Middle Directory Table*)
- tablica stron (*Page Table*)
- przesunięcie (*Offset*) 12 najmniej znaczących bitów

Procesory 32-bitowe używają tylko dwóch rodzajów tablic stron, więc pośredni katalog stron jest eliminowany (zawiera zero bitów). Dzięki przechowywaniu pozycji tego katalogu w sekwencji wskaźników ten sam kod może działać na architekturach 32- i 64-bitowych.[62]

[62] Obsługa bardzo dużych ilości pamięci RAM wymaga zastosowania dodatkowych poziomów stronicowania, np. CentOS 8 używa 5-ciu poziomów, co pozwala zaadresować 128 PB pamięci.

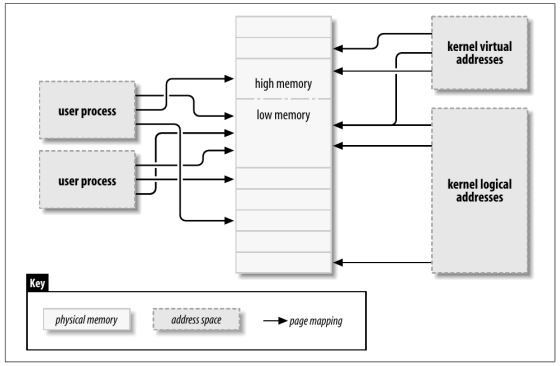
Stronicowanie w procesorach Intela (cd)

Poczynając od procesorów Intel Pentium wprowadzono *rozszerzone stronicowanie*, które umożliwia na stosowanie bloków stronicowych o wielkości 4 KB lub 4 MB.

Przy rozszerzonym stronicowaniu najbardziej znaczących 10 bitów adresu liniowego określa *Directory*, które wskazuje na położenie tablicy stron, a pozostałe 22 bitów wskazuje na przesunięcie, czyli adres fizyczny w ramach 4 MB strony.

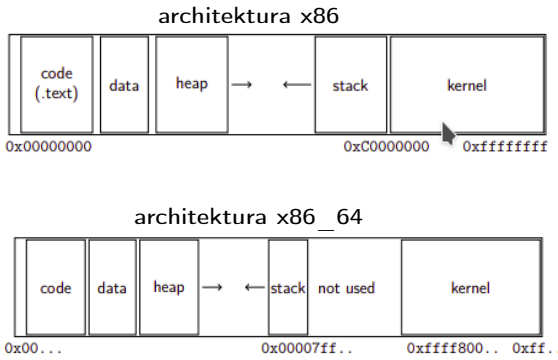
Zastosowanie jednostki PAE (*Physical Address Extension*) pozwala na stosowanie adresów 36-o bitowych, czyli adresowanie do 64 GB pamięci fizycznej. Ponieważ adresy są nadal 32-u bitowe, więc tylko 4 GB pamięci mogą być „stale odwzorowane" i dostępne w danej chwili.

Linux: typy adresów[63]



[63] Linux Device Drivers, Third Edition

Linux: przestrzeń adresowa procesu[64]



[64] Memory handout

Linux: przestrzeń adresowa procesu[65]

- Przestrzeń wirtualna każdego procesu w Linuksie jest podzielona na przestrzeń jądra oraz przestrzeń użytkownika.
- Rozmiar przestrzeni adresowej użytkownika jest określony przez zmienną `TASK_SIZE`: 3 GB (x86) lub 128 TB (x86_64).[66]
- Przestrzeń użytkownika jest związana z procesem (każdy proces ma prywatną przestrzeń adresową), tzn. jest mapowana przez osobną tablicę stron.
- Przestrzeń jądra jest dzielona przez wszystkie procesy poprzez odwzorowywanie jej na górną część przestrzeni adresowej każdego procesu.
- W czasie wykonywania procesu w trybie użytkownika tylko przestrzeń użytkownika jest dostępna.
- W trybie jądra dostępna jest przestrzeń jądra i użytkownika.

[65] Virtual Memory in the IA-64 Linux Kernel
[66] Dla systemów x86_64 przestrzeń adresowa jądra wynosi 128 TB.

Pamięć wirtualna: strefy pamięci w architekturze x86

Strefa	Opis	Pamięć fizyczna
ZONE_DMA	strony nadające się do przesyłu DMA	< 16 MB
ZONE_NORMAL	strony normalnie adresowane	16 – 896 MB
ZONE_DMA32	strony nadające się do przesyłu DMA	896 – 4 GB
ZONE_HIGHMEM	strony odwzorowywane dynamicznie	> 896 MB

Przestrzeń adresowa jądra wynosi 1 GB. Z tego obszaru 128 MB jest przeznaczonych na struktury danych służące do obsługi pamięci wysokiej Dlatego przestrzeń adresowa jądra wynosi 896 MB (1024-128=896 MB).

Nowsze urządzenia wej/wyj (PCI) stosują adresację 32-bitową i dlatego na potrzeby DMA można wykorzystać większy obszar pamięci.[67]

Część procesów jądra nie korzysta z przestrzeni wirtualnej (zob. top).

[67] Kernel development.
Problem 3 GB czy 4 GB RAM: patrz 3GB or 4GB – a 'Virtual' Memory Upgrade

Pamięć wirtualna: strefy pamięci dla x86 i x86_64

Zawartość /proc/buddyinfo:

```
x86
Node 0, zone      DMA     35     42     22     13      2      3     ...
Node 0, zone   Normal   5937   6716   1253    103      2      1     ...
Node 0, zone  HighMem   6460   1165     86      0      0      0     ...

x86_64
Node 0, zone      DMA      1      1      2      1      0      0     ...
Node 0, zone    DMA32    493    153     57     38     20     13     ...
Node 0, zone   Normal   1258   2187   1922   1986   1426   1133     ...
Node 1, zone   Normal   2964   3494   2779   2134   1596   1327     ...
Node 2, zone   Normal    857   2093   1832   1374   1354   1244     ...
Node 3, zone   Normal   1665   2196   2244   1889   1670   1521     ...
```

Obserwacja fragmentacji pamięci: `collectl -sB`


```
$ ./iomem.pl -u m

          0.0000 M          0.0039 M  reserved
          0.0039 M          0.6211 M  System RAM
          0.6211 M          0.6250 M  reserved
          0.6250 M          0.7500 M  Video RAM area
          0.7500 M          0.7812 M  Video ROM
          0.8086 M          0.8125 M  Adapter ROM
          0.9375 M          1.0000 M  System ROM
          1.0000 M       3318.5049 M  System RAM
          4.0000 M          7.7538 M  Kernel code
          7.7538 M         10.2367 M  Kernel data
         10.8008 M         12.3757 M  Kernel bss
       3318.5049 M       3328.0000 M  reserved
       3318.5049 M       3319.0000 M  pnp 00:00
       3319.0000 M       3320.0000 M  pnp 00:00
       3328.0000 M       3584.0000 M  0000:00:02.0
       3584.0000 M       3586.0000 M  PCI Bus 0000:0d
       ...
       4077.2500 M       4077.2695 M  pnp 00:0d
       4077.2695 M       4077.6250 M  pnp 00:00
       4078.0000 M       4078.0625 M  reserved
       4078.0000 M       4078.0625 M  pnp 00:00
       4078.0000 M       4078.0039 M  Local APIC
       4090.5000 M       4090.5010 M  0000:00:1d.7
       4090.5000 M       4090.5010 M  ehci_hcd
       4091.0000 M       4096.0000 M  reserved
       4091.0000 M       4096.0000 M  pnp 00:00
```



```
$ ./iomem.pl -u m | grep PCI

          0.6250 M          0.7500 M  PCI Bus 0000:00
          0.8125 M          0.8750 M  PCI Bus 0000:00
       3328.0000 M       3840.0000 M  PCI Bus 0000:00
       3584.0000 M       3586.0000 M  PCI Bus 0000:0d
       3586.0000 M       3588.0000 M  PCI Bus 0000:0b
       3588.0000 M       3590.0000 M  PCI Bus 0000:0b
       3590.0000 M       3592.0000 M  PCI Bus 0000:0c
       3592.0000 M       3594.0000 M  PCI Bus 0000:09
       3648.0000 M       3712.0000 M  PCI Bus 0000:03
       3648.0000 M       3712.0000 M  PCI CardBus 0000:04
       3833.0000 M       3834.0000 M  PCI Bus 0000:03
       3834.0000 M       3836.0000 M  PCI Bus 0000:0d
       3836.0000 M       3837.0000 M  PCI Bus 0000:09
       3837.0000 M       3838.0000 M  PCI Bus 0000:0c
       3840.0000 M       3904.0000 M  PCI MMCONFIG 0000 [bus 00-3f]
       3904.0273 M       3904.0312 M  PCI Bus 0000:00
       3904.0469 M       4076.0000 M  PCI Bus 0000:00
       3968.0000 M       4032.0000 M  PCI CardBus 0000:04
       4076.0625 M       4077.0000 M  PCI Bus 0000:00
       4077.0010 M       4077.1250 M  PCI Bus 0000:00
       4077.2500 M       4077.2695 M  PCI Bus 0000:00
       4078.0625 M       4091.0000 M  PCI Bus 0000:00
```

```
# pmap.pl  -u k rsyslogd
92677242668.0000 K      84K r---- rsyslogd
92677242752.0000 K     384K r-x-- rsyslogd
92677243136.0000 K     200K r---- rsyslogd
92677243340.0000 K      12K r---- rsyslogd
92677243352.0000 K      28K rw--- rsyslogd
92677243380.0000 K       4K rw---   [ anon ]
92677251460.0000 K     264K rw---   [ anon ]
136430431736.0000 K    8192K r--s- system.journal
...
136430871920.0000 K       4K r---- ld-2.30.so
136430871924.0000 K       4K rw--- ld-2.30.so
136430871928.0000 K       4K rw---   [ anon ]
137431694932.0000 K     132K rw---   [ stack ]
137431695116.0000 K      16K r----   [ anon ]
137431695132.0000 K       8K r-x--   [ anon ]
18014398509471744.0000 K       4K r-x--   [ anon ]


# size `which rsyslogd`
  text    data     bss     dec     hex filename
 677978   39176    4216  721370   b01da /usr/sbin/rsyslogd
```


Fedora 26: dlaczego dwa procesy bash mają różne przestrzenie wirtualne?

```
$ pmap.pl -ug PID-bash1
  87556.7925 G    1044K r-x-- bash
  87556.7955 G      16K r---- bash
  87556.7955 G      36K rw--- bash
  87556.7955 G      40K rw---   [ anon ]
  87556.7960 G     648K rw---   [ anon ]
 130069.4770 G      44K r-x-- libnss_files-2.25.so
 130069.4771 G    2044K ----- libnss_files-2.25.so
 130069.4790 G       4K r---- libnss_files-2.25.so
 130069.4790 G       4K rw--- libnss_files-2.25.so

$ pmap.pl -ug PID-bash2
  87580.7969 G    1044K r-x-- bash
  87580.7999 G      16K r---- bash
  87580.7999 G      36K rw--- bash
  87580.7999 G      40K rw---   [ anon ]
  87580.8054 G     640K rw---   [ anon ]
 130132.0723 G      44K r-x-- libnss_files-2.25.so
 130132.0724 G    2044K ----- libnss_files-2.25.so
 130132.0743 G       4K r---- libnss_files-2.25.so
 130132.0743 G       4K rw--- libnss_files-2.25.so
```


CentOS 7: dlaczego dwa procesy bash mają częściowo różne przestrzenie wirtualne?

```
$ pmap.pl -um PID-bash1
         4.0000 M     884K r-x-- bash
         6.8594 M       4K r---- bash
         6.8633 M      36K rw--- bash
         6.8984 M      24K rw---   [ anon ]
        17.8516 M    1844K rw---   [ anon ]
  134072498.9531 M       8K r-x-- ISO8859-2.so
  134072498.9609 M    2044K ----- ISO8859-2.so
  134072500.9570 M       4K r---- ISO8859-2.so

$ pmap.pl -um PID-bash2
         4.0000 M     884K r-x-- bash
         6.8594 M       4K r---- bash
         6.8633 M      36K rw--- bash
         6.8984 M      24K rw---   [ anon ]
        36.0977 M    1720K rw---   [ anon ]
  133630335.7891 M       8K r-x-- ISO8859-2.so
  133630335.7969 M    2044K ----- ISO8859-2.so
  133630337.7930 M       4K r---- ISO8859-2.so
```


Wiązanie adresów[68]

- kompilacja
  - kompilator wiąże adresy symboliczne z adresami bezwzględnymi lub względnymi (przenaszalnymi, relokowalnymi), tj. adresami liczonymi względem początku danego bloku (modułu) programu
  - kompilator może tworzyć kod wynikowy z adresami bezwzględnymi, np. pliki .COM w systemie MS-DOS
- łączenie
- ładowanie

  ładowanie dynamiczne – podprogram jest wprowadzany do pamięci dopiero wówczas, kiedy jest potrzebny (został wywołany)
- wykonanie – stronicowanie wymaga zamiany adresów wirtualnych na fizyczne w trakcie wykonania programu

[68] W. Stallings, *Operating Systems: Internals and Design Principles, 6/E*

Wiązanie adresów

- łączenie dynamiczne – dotyczy (głównie) bibliotek systemowych (*biblioteki dzielone*) i powoduje odroczenie łączenia do czasu konieczności odwołania się do danej funkcji bibliotecznej (w odpowiednim miejscu kodu programu umieszcza się zakładkę do tej funkcji)
  - wiele programów może korzystać z jednej kopii podprogramu bibliotecznego
  - ułatwione poprawianie błędów systemowych; wadliwe podprogramy biblioteczne można zastępować nowymi wersjami
  - `ldconfig -p`, `ldconfig -v`

Pamięć a efektywność systemu

W celu przyspieszenia działania systemu stosuje się pamięci podręczne (*cache*):

- **buforów** (*buffer cache*) zawierają dane, które są przesyłane z lub do urządzeń blokowych; bufory są indeksowane identyfikatorem urządzenia oraz numerem bloku danych
- **stron** (*page cache*) jest używany do przyspieszenia dostępu do obrazu programu i danych
- **wymiany** (*swap cache*) służy do zapisywania brudnych stron, które muszą być usunięte z pamięci

Od wersji 2.4 jądro Linuksa zawiera ujednoliconą pamięć podręczną stron, tj. pamięć podręczna buforów jest realizowana poprzez pamięć podręczną stron. Bufory opisują odwzorowanie bloku na stronę pamięci, która może znajdować się w pamięci podręcznej stron.

notebook:

```
$ free -mt
              total        used        free      shared  buff/cache   available
Mem:           7690        4785         231         617        2673        1832
Swap:          7823        2907        4916
Total:        15514        7692        5148
```

serwer:

```
$ free -mt -w
              total        used        free      shared     buffers       cache   available
Mem:         257678      176603       42061         516         187       38827       76405
Swap:          9999         304        9695
Total:       267678      176908       51756
```


stacja robocza (x86_32): `/proc/meminfo`

```
MemTotal:       507404 kB        AnonPages:      190600 kB
MemFree:          7752 kB        Mapped:          54368 kB
Buffers:         34392 kB        Slab:            49220 kB
Cached:         158584 kB        PageTables:       4792 kB
SwapCached:        128 kB        NFS_Unstable:        0 kB
Active:         337976 kB        Bounce:              0 kB
Inactive:        45708 kB        CommitLimit:    848096 kB
HighTotal:           0 kB        Committed_AS:   277708 kB
HighFree:            0 kB        VmallocTotal:   516088 kB
LowTotal:       507404 kB        VmallocUsed:      4296 kB
LowFree:          7752 kB        VmallocChunk:   510964 kB
SwapTotal:      594396 kB        HugePages_Total:     0
SwapFree:       594268 kB        HugePages_Free:      0
Dirty:            2804 kB        HugePages_Rsvd:      0
Writeback:           0 kB        Hugepagesize:     4096 kB
```

zob.: `man 5 proc`, /usr/src/linux/Documentation/filesystems/proc.txt

notebook (x86_64, Fedora 27): `/proc/meminfo`

```
MemTotal:        7866196 kB
MemFree:          217384 kB
MemAvailable:    1616344 kB
Buffers:          481128 kB
Cached:           765140 kB
SwapCached:         3148 kB
Active:          5489860 kB
Inactive:         754904 kB
Active(anon):    4850232 kB
Inactive(anon):   704268 kB
Active(file):     639628 kB
Inactive(file):    50636 kB
Unevictable:          48 kB
Mlocked:              48 kB
SwapTotal:       8191992 kB
SwapFree:        7704208 kB
Dirty:               964 kB
Writeback:             0 kB
AnonPages:       4996700 kB
Mapped:           534748 kB
Shmem:            556004 kB
Slab:            1104992 kB
SReclaimable:    1011768 kB
SUnreclaim:        93224 kB
KernelStack:       21904 kB
PageTables:       157544 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    12125088 kB
Committed_AS:   18691604 kB
VmallocTotal:   34359738367 kB
VmallocUsed:           0 kB
VmallocChunk:          0 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:      463736 kB
DirectMap2M:     7620608 kB
DirectMap1G:           0 kB
```


serwer (x86_64, CentOS7): `/proc/meminfo`

```
MemTotal:       263859604 kB
MemFree:        48533248 kB
MemAvailable:   75392736 kB
Buffers:          193188 kB
Cached:         20376848 kB
SwapCached:        46912 kB
Active:         189824552 kB
Inactive:       10517744 kB
Active(anon):   177789928 kB
Inactive(anon):  2328752 kB
Active(file):   12034624 kB
Inactive(file):  8188992 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:      10239996 kB
SwapFree:       10148860 kB
Dirty:              484 kB
Writeback:             0 kB
AnonPages:      179728716 kB
Mapped:           47028 kB
Shmem:              346424 kB
Slab:           11536992 kB
SReclaimable:    7520032 kB
SUnreclaim:      4016960 kB
KernelStack:       13328 kB
PageTables:       499436 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    142169796 kB
Committed_AS:   302559624 kB
VmallocTotal:   34359738367 kB
VmallocUsed:      1467316 kB
VmallocChunk:   34224007224 kB
HardwareCorrupted:     0 kB
AnonHugePages:   2234368 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:      288556 kB
DirectMap2M:    22681600 kB
DirectMap1G:    247463936 kB
```


Sprzętowa pamięć podręczna[69]

- Przy współczesnych procesorach gigaherzowych trzeba kilkudziesięciu cykli zegara w celu dokonania zapisu/odczytu pamięci DRAM.
- Sprzętowe pamięci podręczne łagodzą dysproporcje w szybkości działania procesora i pamięci operacyjnej. Ich działanie opiera się na lokalności, którą wykazują programy i struktury danych (wynika to z cyklicznej natury programów i ułożenia danych w tablicach).
- Sprzętowa pamięć podręczna, która jest mniejsza od pamięci operacyjnej, lecz od niej szybsza, zawiera ostatnio używany kod i dane.
- W architekturze Intela utworzono nową jednostkę o nazwie *linia*. Składa się ona z kilkudziesięciu ciągłych bajtów (2^6), które są przesyłane w jednej wiązce pomiędzy powolną pamięcią operacyjną (DRAM), a szybkimi układami statycznej pamięci RAM (SRAM), używanymi do budowy pamięci podręcznej.

[69] Ulrich Drepper, Memory part 2: CPU caches

Sprzętowa pamięć podręczna

Sterownik pamięci podręcznej przechowuje informacje o każdej linii umieszczonej w pamięci podręcznej (linie są grupowane w oddzielne podzbiory). Przy odwołaniu do komórki pamięci RAM, procesor pobiera z adresu fizycznego indeks podzbioru i porównuje znaczniki wszystkich linii w podzbiorze z najbardziej znaczącymi bitami adresu fizycznego.

W wyniku porównania zachodzi

- trafienie (*cache hit*)
- chybienie (*cache miss*)

Trafienie przy operacji odczytu oznacza, że sterownik pobiera dane z linii i przenosi je z pamięci podręcznej do rejestru procesora; procesor nie korzysta wówczas z pamięci RAM i zyskuje na czasie.

Sprzętowa pamięć podręczna (cd)

Przy operacji zapisu sterownik stosuje strategię

- bezpośredniego zapisu (*write-through*), tj. sterownik modyfikuje zapis w pamięci podręcznej i pamięci głównej
- zapisu z opóźnieniem (*write-back*), tj. sterownik modyfikuje zapis w linii pamięci podręcznej; modyfikacja pamięci RAM następuje przy usuwaniu linii z pamięci podręcznej

Systemy wieloprocesorowe zawierają oddzielne sprzętowe pamięci podręczne dla każdego procesora. Potrzebny jest zatem dodatkowy układ sprzętowy do synchronizowania zawartości tych pamięci.

Pierwsze modele Pentium zawierały tylko pamięć podręczną pierwszego poziomu (L1-cache). Nowsze modele zawierają dodatkowy, wolniejszy układ L2-cache. Zachowanie spójności pomiędzy tymi dwoma pamięciami jest realizowane na poziomie sprzętowym.

Bufory translacji adresów

Stronicowanie oznacza stałą zamianę adresów liniowych na fizyczne przy wykorzystaniu katalogów i tablic stron, które znajdują się w (wolnej) pamięci RAM.

Współczesne procesory umożliwiają przyspieszenie tej czynności dzięki wykorzystaniu pamięci podręcznej zwanej *buforami translacji adresów* (TLB, *Translation Lookaside Buffers*).

TLB – specjalny zestaw specjalnych rejestrów (od 8 do 2048), tzw. rejestrów asocjacyjnych; każdy zawiera klucz (adres strony) i wartość (adres ramki).

Adres fizyczny jest obliczany za pomocą tablic stron tylko przy pierwszym odwołaniu. Następne odwołania korzystają z tego adresu przechowywanego w TLB (narzut rzędu 10%).

Czym jest plik?

- plik jest zbiorem powiązanych ze sobą informacji, zdefiniowanych przez jego twórcę
- w plikach przechowuje się programy oraz dane (dane mogą być liczbowe, tekstowe, alfanumeryczne)
- format plików może być dowolny lub ściśle określony
- zawartość plików można traktować jako ciąg bitów, bajtów, wierszy lub rekordów

Zarządzanie plikami

Dla wygody użytkownika system operacyjny:

- tworzy logicznie jednolity obraz magazynowania informacji
- definiuje pliki, czyli jednostki logiczne przechowywania informacji, niezależne od fizycznych własności użytych urządzeń pamięci
- umożliwia tworzenie i usuwanie plików
- dostarcza elementarnych operacji do manipulowania plikami
- odwzorowuje pliki na obszary pamięci pomocniczej
- odwzorowuje pliki na urządzenia fizyczne
- składuje pliki na trwałych nośnikach pamięci

Rodzaje plików

- zwykłe pliki (ASCII lub binarne)
- katalogi
- dowiązania symboliczne
- urządzenia blokowe (modelujące dyski)
- urządzenia znakowe (modelujące terminale, drukarki, adaptery sieciowe, modemy, itp.)
- potoki nazwane (*named pipes*, łącza nazwane, kolejki FIFO) – specjalne pliki łączące procesy
- gniazda

Struktura plików w Linuksie

Komenda: `ls -F`

```
backup/  ccna/   etc/     HOWTOS@  linux-doc/   mnt/   root/
bin/     cdrom@  floppy@  initrd/  lost+found/  opt/   sbin/
usr/     boot/    dev/    home/    lib/     misc/      proc/
tmp/     var/
```

Komenda: `df -T`

```
Filesystem    Type   1K-blocks      Used Available Use% Mounted on
/dev/hda2     ext3     2925332   2476332    389560  87% /
/dev/hda4     ext3     1050328    956308     72340  93% /home
none         tmpfs      127836         0    127836   0% /dev/shm
/dev/sda3     ext3    20648448   5187560  14412008  27% /usb-home
```


Nazwy plików:

- bezwzględne – ścieżka nazwy pliku zaczyna się w korzeniu struktury plików

  np. /etc/sysc/config/hwconfig

- względne – ścieżka nazwy pliku zaczyna się w bieżącym (roboczym) katalogu

  np. ../spool/mail

Własności plików

Komenda: `ls -la`

```
drwxr-xr-x  31 note     kmk     3072 Feb  4 12:31 .
drwxr-xr-x  10 root     kmk     1024 Mar 19  1998 ..
-rw-r--r--   1 note     kmk      977 Jan 27 18:07 .history
-rw-rw-r--   1 note     kmk      625 Dec 15  1997 .login
-rw-rw-r--   1 note     kmk       51 Nov 17  1997 .logout
-rw-rw-r--   1 note     kmk     1410 Feb 13  1998 .tcshrc
drwxr-xr-x   7 note     kmk     1024 Jan  5 21:06 2dhf_dist
drwx------   2 note     kmk     1024 Mar 20  1998 Mail
```

- *discretionary access control* (DAC) – uznaniowa kontrola dostępu (Unix/Linux)
- *mandatory access control* (MAC) – obowiązkowa kontrola dostępu, np. SELinux (Linux), Mandatory Integrity Control (Windows Vista)

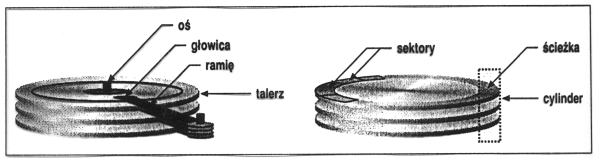
Własności plików

		plik	katalog
r	4	prawo odczytu pliku	prawo przeglądania zawartości
w	2	prawo zapisu pliku	prawo tworzenia plików
x	1	prawo wykonania pliku	prawo dostępu do katalogu

```
700 = rwx------ = u+rwx
711 = rwx--x--x = u+rwx,go+x
755 = rwxr-xr-x = u=rwx,go=rx
1755 = rwxr-xr-t = u=rwx,go=rx,+t
2511 = r-x--s--x = u=rx,g=xs,o=x
4511 = r-s--x--x = u=rxs,go=x
```

```
-r-s--x--x    1 root     root        15368 May 28  2002 /usr/bin/passwd

drwxrwxrwt   36 root     root         4096 mar 21 19:15 tmp
```


Cylindry, głowice, sektory (CHS)[70]




```
Disk /dev/fd0: 1 MB, 1474560 bytes
2 heads, 18 sectors/track, 80 cylinders
Units = cylinders of 36 * 512 = 18432 bytes
```

[70] http://www.thomas-krenn.com/en/wiki/Advanced_Sector_Format_of_Block_Devices

Struktura danych blokowych: sectory, partycje[71]

```
# parted /dev/sda print
Model: ATA FUJITSU MHV2080A (scsi)
Disk /dev/sda: 80.0GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
Number  Start   End     Size    Type     File system  Flags
 1      1049kB  1075MB  1074MB  primary  ext4         boot
 2      1075MB  80.0GB  79.0GB  primary               lvm

# fdisk -l  /dev/sda
Disk /dev/sda: 74.5 GiB, 80026361856 bytes, 156301488 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xeb0cedd2
Device     Boot   Start       End   Sectors  Size Id Type
/dev/sda1  *       2048   2099199   2097152    1G 83 Linux
/dev/sda2       2099200 156301311 154202112 73.5G 8e Linux LVM
```

[71] http://www.thomas-krenn.com/en/wiki/Advanced_Sector_Format_of_Block_Devices

Struktura danych blokowych: sectory, partycje

```
[root@sr1-0 ~]# parted /dev/sdc print
Model: AVAGO SMC3108 (scsi)
Disk /dev/sdc: 16.0TB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  16.0TB  16.0TB  ext4         primary  lvm

-------------------------------------------------------------------------------

[root@sr1-0 ~]# parted /dev/sdc unit TiB print
Model: AVAGO SMC3108 (scsi)
Disk /dev/sdc: 14.6TiB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number  Start    End      Size     File system  Name     Flags
 1      0.00TiB  14.6TiB  14.6TiB  ext4         primary  lvm
```

Struktura danych blokowych: sectory, partycje

```
[root@sr1-0 ~]# fdisk  -l /dev/sdc
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase. \
                                                        Use at your own discretion.
Disk /dev/sdc: 16000.9 GB, 16000900661248 bytes, 31251759104 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk label type: gpt
Disk identifier: DE3CEFB7-A858-427A-AE9A-86FAE0EFC6B3
#         Start          End    Size  Type            Name
 1         2048  31251757055   14.6T  Linux LVM       primary
```

```
[root@sr1-0 ~]# parted /dev/sdc unit s print
Model: AVAGO SMC3108 (scsi)
Disk /dev/sdc: 31251759104s
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:
Number  Start  End           Size          File system  Name     Flags
 1      2048s  31251757055s  31251755008s  ext4         primary  lvm
```


Struktura danych blokowych: sectory, partycje

```
root@scobie ~$ fdisk -l /dev/sda
Disk /dev/sda: 465.8 GiB, 500107862016 bytes, 976773168 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 8B305206-A529-48D8-A58F-28CFF18C7853

Device     Start       End   Sectors   Size Type
/dev/sda1   2048      4095      2048     1M BIOS boot
/dev/sda2   4096 976773119 976769024 465.8G Linux LVM
```

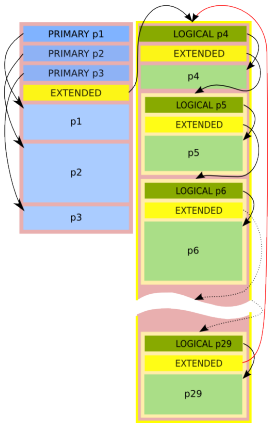
```
root@scobie ~$ parted /dev/sda print
Model: ATA Samsung SSD 850 (scsi)
Disk /dev/sda: 500GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  2097kB  1049kB               primary  bios_grub
 2      2097kB  500GB   500GB                primary  lvm
```


Struktura danych blokowych: partycje

- tabela partycji MSDOS
  - pierwszy sektor dysku (sektor 0, MBR (*Master Boot Record*) zawiera kod programu do ładowania systemu operacyjnego oraz tabelę partycji (64 B)
  - do 4 partycji typu podstawowego (*primary*)
  - partycja rozszerzona (*extended*) typu 0x5 (dostęp via CHS), 0xF (dostęp via LBA) lub 0x85: partycja podstawowa podzielona na partycje logiczne; adres pierwszego sektora partycji logicznej zawiera tabelę partycji, która definiuje partycję logiczną i kolejną partycję rozszerzoną[72]
- tabela partycji GPT (*Globally Unique Identifier Partition Table*)
  - dowolna liczba partycji (domyślnie 128)
  - położenie: sektory 1-32

[72] List of partition identifiers for PCs

Partycje podstawowe, rozszerzone i logiczne[73]



[73] http://tjworld.net/wiki/Android/HTC/Vision/EmmcPartitioning

Identyfikatory partycji wg fdisk (tabela MSDOS)

0	Empty	24	NEC DOS	81	Minix / old Lin	bf	Solaris
1	FAT12	27	Hidden NTFS Win	82	Linux swap / So	c1	DRDOS/sec (FAT-
2	XENIX root	39	Plan 9	83	Linux	c4	DRDOS/sec (FAT-
3	XENIX usr	3c	PartitionMagic	84	OS/2 hidden or	c6	DRDOS/sec (FAT-
4	FAT16 <32M	40	Venix 80286	85	Linux extended	c7	Syrinx
5	Extended	41	PPC PReP Boot	86	NTFS volume set	da	Non-FS data
6	FAT16	42	SFS	87	NTFS volume set	db	CP/M / CTOS / .
7	HPFS/NTFS/exFAT	4d	QNX4.x	88	Linux plaintext	de	Dell Utility
8	AIX	4e	QNX4.x 2nd part	8e	Linux LVM	df	BootIt
9	AIX bootable	4f	QNX4.x 3rd part	93	Amoeba	e1	DOS access
a	OS/2 Boot Manag	50	OnTrack DM	94	Amoeba BBT	e3	DOS R/O
b	W95 FAT32	51	OnTrack DM6 Aux	9f	BSD/OS	e4	SpeedStor
c	W95 FAT32 (LBA)	52	CP/M	a0	IBM Thinkpad hi	ea	Rufus alignment
e	W95 FAT16 (LBA)	53	OnTrack DM6 Aux	a5	FreeBSD	eb	BeOS fs
f	W95 Ext'd (LBA)	54	OnTrackDM6	a6	OpenBSD	ee	GPT
10	OPUS	55	EZ-Drive	a7	NeXTSTEP	ef	EFI (FAT-12/16/
11	Hidden FAT12	56	Golden Bow	a8	Darwin UFS	f0	Linux/PA-RISC b
12	Compaq diagnost	5c	Priam Edisk	a9	NetBSD	f1	SpeedStor
14	Hidden FAT16 <3	61	SpeedStor	ab	Darwin boot	f4	SpeedStor
16	Hidden FAT16	63	GNU HURD or Sys	af	HFS / HFS+	f2	DOS secondary
17	Hidden HPFS/NTF	64	Novell Netware	b7	BSDI fs	fb	VMware VMFS
18	AST SmartSleep	65	Novell Netware	b8	BSDI swap	fc	VMware VMKCORE
1b	Hidden W95 FAT3	70	DiskSecure Mult	bb	Boot Wizard hid	fd	Linux raid auto
1c	Hidden W95 FAT3	75	PC/IX	bc	Acronis FAT32 L	fe	LANstep
1e	Hidden W95 FAT1	80	Old Minix	be	Solaris boot	ff	BBT

Typy partycji wg fdisk (tabela GPT)

```
...
11 Microsoft basic data                  EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
12 Microsoft LDM metadata                5808C8AA-7E8F-42E0-85D2-E1E90434CFB3
13 Microsoft LDM data                    AF9B60A0-1431-4F62-BC68-3311714A69AD
14 Windows recovery environment          DE94BBA4-06D1-4D40-A16A-BFD50179D6AC
15 IBM General Parallel Fs               37AFFC90-EF7D-4E96-91C3-2D7AE055B174
16 Microsoft Storage Spaces              E75CAF8F-F680-4CEE-AFA3-B001E56EFC2D
...
19 Linux swap                            0657FD6D-A4AB-43C4-84E5-0933C84B4F4F
20 Linux filesystem                      0FC63DAF-8483-4772-8E79-3D69D8477DE4
21 Linux server data                     3B8F8425-20E0-4F3B-907F-1A25A76F98E8
22 Linux root (x86)                      44479540-F297-41B2-9AF7-D131D5F0458A
23 Linux root (ARM)                      69DAD710-2CE4-4E3C-B16C-21A1D49ABED3
24 Linux root (x86-64)                   4F68BCE3-E8CD-4DB1-96E7-FBCAF984B709
25 Linux root (ARM-64)                   B921B045-1DF0-41C3-AF44-4C6F280D3FAE
26 Linux root (IA-64)                    993D8D3D-F80E-4225-855A-9DAF8ED7EA97
27 Linux reserved                        8DA63339-0007-60C0-C436-083AC8230908
28 Linux home                            933AC7E1-2EB4-4F13-B844-0E14E2AEF915
29 Linux RAID                            A19D880F-05FC-4D3B-A006-743F0F84911E
30 Linux extended boot                   BC13C2FF-59E6-4262-A352-B275FD6F7172
31 Linux LVM                             E6D6D379-F507-44C2-A23C-238F2A3DF928
...
```


Struktura danych blokowych: nagrywanie ZBR

- We współczesnych dyskach nie stosuje się podziału powierzchni dysku na jednakową liczbę sektorów. Dysk dzieli się na pewną liczbę (zwykle 15) stref; im strefa leży bliżej krawędzi zewnętrzej tym ma więcej sektorów. Zastosowanie tzw. nagrywania ZBR (*Zone Bit Rate*) powoduje zapisywanie danych z prawie jednakową gęstością (wyrażaną w bitach na jednostkę długości ścieżki).
- Szybkość zapisu/odczytu danych zależy od ich położenia na dysku.
- Cylinder o najmniejszym numerze jest najszybszy, największy i najbardziej gęsty (najwięcej bitów/cylinder).
- Cylinder o największym numerze jest najwolniejszy, najmniejszy i najmniej gęsty.

Systemy plików

- System plików jest odpowiedzialny za zorganizowanie plików w logiczne hierarchiczne struktury składające się z katalogów, zwykłych plików, dowiązań symbolicznych (miękkich), itp.
- Pliki, itd. są przechowywane w blokach na urządzeniu fizycznym.
- Uniksowy/Linuksowy system plików traktuje to urządzenie jako liniowy zbiór bloków.
- Dla systemu plików fizyczny układ bloków na dysku jest bez znaczenia. Zadaniem sterownika urządzenia blokowego jest powiązanie numeru bloku z numerem cylindra, głowicy, sektora.

Systemy plików

Tworzenie partycji: `parted, gdisk, fdisk`
Tworzenie systemu plików: `mkfs`

```
# mkfs -t ext4 /dev/sda2
# mkfs -t vfat -c -F 32 -I /dev/sdc
```

```
# df -Ti
Filesystem     Type    Inodes   IUsed   IFree IUse% Mounted on
/dev/sda2      ext4    371680  150841  220839   41% /
/dev/sda1      ext4    271296   31520  239776   12% /home
none          tmpfs     31959       1   31958    1% /dev/shm
/dev/loop0     vfat          0       0       0     - /mnt/zero
```

ext2|3|4: *Second|Third|Fourth Extended File System*
tmpfs: /usr/src/linux-$(uname -r)/Documentation/filesystems/tmpfs.txt

Systemy plików

- dyskowe systemy plików
  - Unix: UFS (*Unix File System*, aka *Berkeley Fast File System*)
  - Linux: ext2|3|4, Btrfs, XFS, JFS, ReiserFS
  - Microsoft: FAT (MSDOS), VFAT (W98), NTFS (WNT)
  - IBM: HPFS (*High Performance File System*)
  - Apple: HFS, HPS+ (*Hierarchical File System*)
  - CD: ISO9660
- sieciowe systemy plików:
  - Unix/Linux: *Network File System* (NFS)
  - Microsoft Windows: *Server Message Block/Common Internet File System* (SMB/CIFS)
  - Novell: *Netware Core Protocol* (NCP)

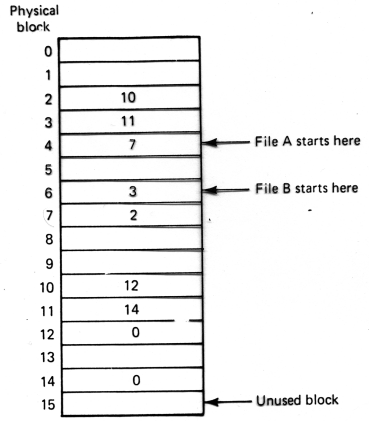
Realizacja systemów plików

- przydział ciągły
  - łatwa lokalizacja pliku
  - bardzo dobra wydajność zapisu/odczytu
  - fragmentacja zewnętrzna
  - trudność w przydziale wolnego obszaru dla plików o zmiennej długości
- połączona lista
  - łatwa lokalizacja pliku
  - brak fragmentacji zewnętrznej
  - mała wydajność odczytu, wielkość pliku nie jest potęgą 2

Realizacja systemów plików

- połączona lista z indeksem (FAT, *File Allocation Table*)
  - plik jest wielokrotnością jednostki alokacji, tzw. klastra
  - brak fragmentacji zewnętrznej
  - wielkość klastra zależy od dostępnej przestrzeni dyskowej
  - dobra wydajność czytania, jeśli FAT w RAM-e
  - duże dyski wymagają dużo RAM-u (około 37 MB dla dysku 10 GB)
- i-węzły
  - łatwa lokalizacja pliku
  - brak fragmentacji zewnętrznej
  - stały rozmiar i-węzła (niezależny od wielkości pliku)
  - dobra wydajność czytania, jeśli i-węzeł w RAM-e
  - dowolna wielkość plików

Realizacja systemów plików: FAT



Pliki w systemie GNU/Linux

- każdy plik jest elementem jakiegoś katalogu
- katalog jest specjalnym rodzajem pliku, który zawiera listę nazw plików wraz z odpowiadającymi im numerami i-węzłów (*i-nodes*)
- i-węzeł jest strukturą danych zawierającą wszystkie dane o pliku (katalogu), m.in. identyfikator właściciela pliku, prawa dostępu, czasy dostępu i zmiany danych i metadanych, numery bloków danych; rozmiar: 128 B (ext3), 256 B (ext4)
- ext3 był domyślnym systemem plików od roku 2000
- ext4 jest domyślnym systemem plików od 2010 (RHEL 8: XFS)
- ext4 vs Btrfs?

Problem: jak w strukturze o małym, stałym rozmiarze pomieścić informację o wszystkich blokach danych pliku niezależnie od jego wielkości?

Systemy plików: montowanie

Przed użyciem systemu plików trzeba wykonać dwie podstawowe operacje:

- rejestrowanie – wykonywane albo w chwili startu systemu, albo w czasie ładowania modułu realizującego system plików; po zarejestrowniu systemu plików jego funkcje stają się dostępne dla jądra
- montowanie – dzięki odpowiednim funkcjom system plików może być zamontowany, czyli udostępniony w drzewie katalogów

```
# mount [-t ext3]               /dev/sda2                     /home
  mount -t <typ systemu plików> <nazwa urządzenia blokowego> <punkt montowania>
```


Systemy plików: montowanie

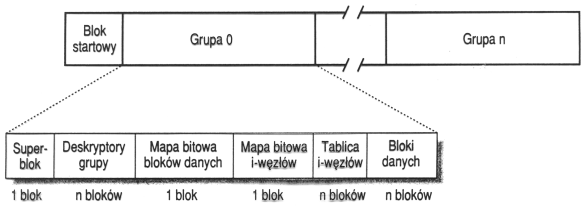
```
# df -hT
Filesystem              Type      Size  Used Avail Use% Mounted on
devtmpfs                devtmpfs  3.8G     0  3.8G   0% /dev
/dev/mapper/fedora-root ext4       26G   17G  8.7G  66% /
tmpfs                   tmpfs     3.8G   37M  3.8G   1% /tmp
/dev/mapper/fedora-home ext4      433G  421G   12G  98% /home
...


# df -hTi
Filesystem              Type     Inodes IUsed IFree IUse% Mounted on
devtmpfs                devtmpfs   953K   577  952K    1% /dev
/dev/mapper/fedora-root ext4       1.7M  246K  1.4M   15% /
tmpfs                   tmpfs      958K   107  958K    1% /tmp
/dev/mapper/fedora-home ext4        28M  1.9M   26M    7% /home
```

System plików extN, N=2,3,4

Elementy systemu plików:

- blok startowy
- grupy bloków
  - superblok
  - deskryptory grupy
  - mapa bitowa bloków danych
  - mapa bitowa i-węzłów
  - tablica i-węzłów
  - bloki danych

System plików extN



System plików extN

```
# dumpe2fs /dev/fedora/root
Group 0: (Blocks 0-32767) csum 0xf7c5 [ITABLE_ZEROED]
  Primary superblock at 0, Group descriptors at 1-3
  Reserved GDT blocks at 4-1027
  Block bitmap at 1028 (+1028)
  Inode bitmap at 1044 (+1044)
  Inode table at 1060-1571 (+1060)
  17207 free blocks, 0 free inodes, 2387 directories
  Free blocks: 15049-15359, 15872-32767
  Free inodes:
Group 1: (Blocks 32768-65535) csum 0x13d3 [ITABLE_ZEROED]
  Backup superblock at 32768, Group descriptors at 32769-32771
  Reserved GDT blocks at 32772-33795
  Block bitmap at 1029 (bg #0 + 1029)
  Inode bitmap at 1045 (bg #0 + 1045)
  Inode table at 1572-2083 (bg #0 + 1572)
  0 free blocks, 0 free inodes, 2741 directories
  Free blocks:
  Free inodes:
Group 2: (Blocks 65536-98303) csum 0xdc6f [ITABLE_ZEROED]
  Block bitmap at 1030 (bg #0 + 1030)
  Inode bitmap at 1046 (bg #0 + 1046)
  Inode table at 2084-2595 (bg #0 + 2084)
  0 free blocks, 0 free inodes, 1144 directories
  Free blocks:
  Free inodes:
```

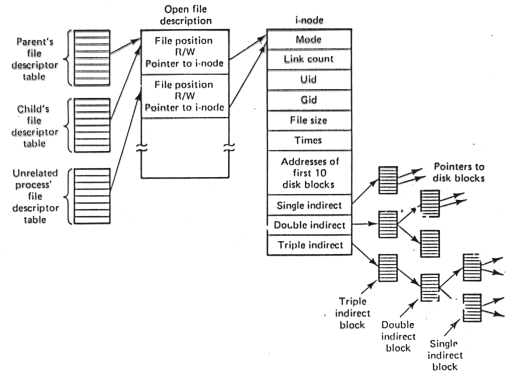

System plików extN: superblok[74]

- struktura danych opisująca globalne parametry systemu plików
- wczytywany do pamięci operacyjnej przy montowaniu systemu plików
- okresowo aktualizowany
- przechowywany w szeregu kopiach

[74] David Both, *An introduction to Linux's EXT4 filesystem*,
Mete Balci, *A Minimum Complete Tutorial of Linux ext4 File System*

```
# tune2fs -l /dev/mapper/fedora-home
tune2fs 1.45.5 (07-Jan-2020)
Filesystem volume name:   <none>
Last mounted on:          /home
Filesystem UUID:          a4498982-6164-4180-a932-7c20392ec1dd
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype needs_recovery extent \
                          64bit flex_bg sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
...
Inode count:              28835840
Block count:              115343360
Reserved block count:     0
Free blocks:              3029509
Free inodes:              26938683
First block:              0
Block size:               4096
Fragment size:            4096
Group descriptor size:    64
Reserved GDT blocks:      1024
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
...
Journal inode:            8
First orphan inode:       9702355
...
```


System plików extN: i-węzły



System plików extN: i-węzły

Niektóre pola i-węzła zwykłego pliku dyskowego ext2|3 (128 B):

- typ pliku i prawa dostępu
- identyfikator właściciela
- długość pliku w bajtach
- czas ostatniego dostępu do pliku
- czas ostatniej zmiany i-węzła
- czas ostatniej modyfikacji pliku
- licznik sztywnych dowiązań
- liczba bloków danych pliku
- wskaźniki do bloków danych

System plików extN: i-węzły

Każdy i-węzeł zawiera

- 12 numerów bloków bezpośrednich
- 3 numery bloków pośrednich

Blok 1 KB = 256 4-bajtowych numerów bloków

Maksymalna wielkość pliku (12 numerów bloków bezpośrednich):

- plus blok pojedynczo pośredni: pliki $\leqslant$ 268 KB (12 KB + 256 KB)
- plus blok podwójnie pośredni:
  pliki $\leqslant$ 64 MB + 268 KB (256×256 KB + 268 KB)
- plus blok pośredni: pliki $\leqslant$ 16 GB + 64 MB +268 KB

Wskaźnik pozycji w pliku (w systemach 32 bitowych) jest 32-bitową liczbą ze znakiem, więc rozmiar plików jest ograniczony do 2 GB (2^{31}-1).

Standard *Large File Support* zwiększa to ograniczenie do 8192 PB (2^{63}-1).[75]

[75] Zob. działanie komendy `getconf FILESIZEBITS <path>` na systemach x86_32 i x86_64.

System plików extN: i-węzły

- katalog – i-węzeł zawiera listę nazw plików (w tym katalogów) wraz z odpowiadającymi im numerami i-węzłów

  Pola pozycji katalogu:

  – numer i-węzła
  – długość pozycji katalogu
  – długość nazwy pliku
  – typ pliku
  – nazwa pliku
- dowiązania symboliczne – i-węzeł zawiera ścieżkę dowiązania symbolicznego ($\leqslant$ 60/255 znaków dla ext3|4)
- pliki urządzeń, potoki nazwane i gniazda – nie wymagają bloków danych

Pliki zwykłe, dowiązania twarde, symboliczne

- informacja o pliku: `stat /etc/passwd`

```
  File: /etc/passwd
  Size: 4291            Blocks: 16         IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 407947      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2021-01-21 17:58:01.014494371 +0100
Modify: 2020-08-28 09:17:41.668819569 +0200
Change: 2020-08-28 09:17:41.670819575 +0200
 Birth: 2020-08-28 09:17:41.668819569 +0200
```

- dowiązanie twarde: `ln <stary plik> <nowy plik>`
- dowiązanie symboliczne: `ln -s <stary plik> <nowy plik>`

System plików ext4[76]

Item	1KiB	2KiB	4KiB	64KiB
Blocks	2^64	2^64	2^64	2^64
Inodes	2^32	2^32	2^32	2^32
File System Size	16ZiB	32ZiB	64ZiB	1YiB
Blocks Per Block Group	8,192	16,384	32,768	524,288
Inodes Per Block Group	8,192	16,384	32,768	524,288
Block Group Size	8MiB	32MiB	128MiB	32GiB
Blocks Per File, Extents	2^32	2^32	2^32	2^32
Blocks Per File, Block Maps	16,843,020	134,480,396	1,074,791,436	4,398,314,962,956 (really 2^32 due to field size limitations)
File Size, Extents	4TiB	8TiB	16TiB	256TiB
File Size, Block Maps	16GiB	256GiB	4TiB	256TiB

[76] ext4 Data Structures and Algorithms

System plików EXT4: zalety

- maksymalny rozmiar pliku: 16 TiB (dla bloków 4 KiB, EXT3 – 2 TiB)
- maksymalny rozmiar systemu plików: 1 EiB = 1024 PiB (EXT3 – 16 TiB)
- katalog może zawierać dowolną liczbę podkatalogów (EXT3 – 32 tys.)
- można montować EXT2|3 jako EXT4
- dodatkowe, nowe cechy:
  – alokacja wieloblokowa; rozmiar *extent*-u do 128 MB ciągłej przestrzeni dyskowej w blokach 4 KB (do czterech numerów ekstentów w i-węźle)
  – trwała prealokacja (gwarancja przydziału określonego, zwykle ciągłego obszaru dysku)
  – alokacja opóźniona, alokator wieloblokowy
  – szybki fsck

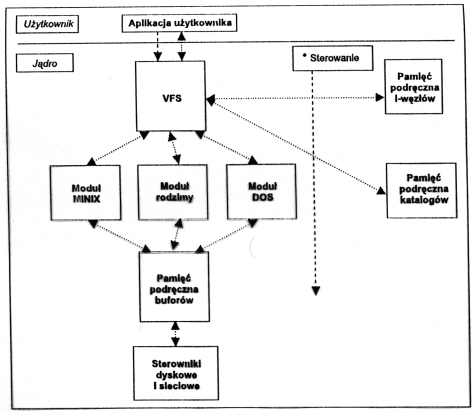
Wirtualny system plików

System operacyjny Linux potrafi obsłużyć kilkanaście różnych systemów plików dzięki zastosowaniu tzw. wirtualnego systemu plików (VFS, *Virtual File System/Switch*).

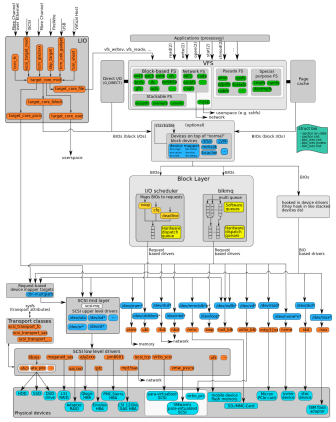
VFS stanowi warstwę interfejsu pomiędzy systemem operacyjnym i różnymi systemami plików.

Podsystem plikowy jądra Linuksa ma bardzo abstrakcyjne pojęcie o systemie plików. Wywołania systemowe `open()`, `read()`, `write()` działają niezależnie od struktury systemu plików i od fizycznych cech nośnika.

Wirtualny system plików



Wirtualny system plików[77]



[77] Virtual file system

Pamięć a efektywność systemu

W celu przyspieszenia działania systemu stosuje się pamięci podręczne:

- **buforów** (*buffer cache*)
- **stron** (*page cache*)
- **wymiany** (*swap cache*)
- **i-węzłów** (tablica i-węzłów)
  - liczba węzłów jest zbyt duża, żeby można je było wczytać wszystkie do pamięci
  - wczytywane są tylko potrzebne i-węzły i umieszczane są w tablicy i-węzłów

Pamięć a efektywność systemu

Jaki i-węzeł opisuje plik gcc?

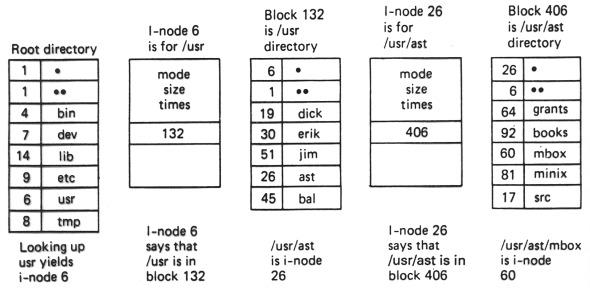
```
# ls -i
81394 /usr/bin/gcc
```

Jak można dotrzeć do tego i-węzła?

```
# ls -i / | grep usr
307042 usr
# ls -i /usr | grep bin
80801 bin
# ls -i /usr/bin | grep gcc
81394 gcc
```

W celu przyspieszenia wyszukiwania nazw Linux używa pamięci podręcznej katalogów (*direcory cache*), która jest częścią podsystemu VFS.

Wyszukiwanie plików



Wirtualne urządzenia blokowe

Wirtualne urządzenia blokowe (`/dev/loop0`, `/dev/loop1`, `...`) pozwalają na montowanie systemów plików, które znajdują się w zwykłych plikach.

```
# losetup /dev/loop0 puppyLinux.iso
# mount /dev/loop0 /media/pup
```

Zamiast dwóch powyższych komend można użyć komendy

```
# mount -o loop puppyLinux.iso /media/pup
```


Specjalne systemy plików: /proc i /sys

Pseudosystemy plików /proc i /sys to interfejsy do struktur danych jądra systemu. Umożliwiają one odczytywanie i zmianę zawartości przestrzeni adresowej innego procesu oraz wykonywanie operacji sterujących tym procesem za pomocą standardowego interfejsu systemu plików.

Komenda `sysctl` pozwala określać oraz zmieniać parametry pracy systemu w czasie jego działania.

```
# sysctl -a
...
m.mmap_rnd_compat_bits = 8
vm.nr_hugepages = 0
vm.nr_hugepages_mempolicy = 0
vm.nr_overcommit_hugepages = 0
vm.nr_pdflush_threads = 0
vm.numa_zonelist_order = Node
vm.oom_dump_tasks = 1
vm.oom_kill_allocating_task = 0
...
```

Bezpośrednie (surowe) wejście-wyjście

„Surowe" (*raw*) urządzenie wejścia-wyjścia:

- specjalny rodzaj urządzenia znakowego, które jest powiązane z urządzeniem blokowym, umożliwiające dostęp do urządzenie bez udziału warstwy buforującej
- operacje wejścia/wyjścia wykonywane są bezpośrednio na urządzeniu (proces bezpośrednio kontaktuje się z urządzeniem)
- program `raw` wiąże surowe urządzenie z urządzeniem blokowym

Zastosowania: systemy zarządzania relacyjnymi bazami danych wykorzystują surowe urządzenia blokowe, ponieważ stosują własne strategie ryglowania (blokowania) plików oraz ze względu na zwiększenie wydajności poprzez optymalizuje operacji wejścia-wyjścia.

Linux oferuje flagę `O_DIRECT`, która umożliwia otwarcie pliku w trybie surowym.

Ograniczenia tradycyjnych systemów plików

- wydajność
- usuwanie skutków awarii
- bezpieczeństwo
- rozmiar

Ograniczenia wydajności systemu dyskowego

- procesor jest dużo szybszy niż dysk (oczekiwanie na zakończenie operacji dyskowych)
- aktywane partycje powinny znajdować się na oddzielnych dyskach
- diagnostyka działania systemu I/O: programy sar, iostat, vmstat, dstat

```
# iostat -d sda2 -xm
Device:  rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz
sda2       0,00 21791,00    1,00  170,00     0,00    85,00  1018,06    94,02

                                                     await  svctm  %util
                                                    512,92   5,85 100,00


# top
top - 11:53:16 up 185 days, 16:16,  2 users,  load average: 3.07, 4.13, 4.43
Tasks: 460 total,   1 running, 459 sleeping,   0 stopped,   0 zombie
%Cpu(s):  8.4 us,  1.6 sy,  0.0 ni, 89.8 id,  20.1 wa,  0.0 hi,  0.1 si,  0.0 st
KiB Mem : 65694968 total,  6400120 free, 13888524 used, 45406324 buff/cache
KiB Swap: 10239996 total,  2456700 free,  7783296 used. 51131164 avail Mem
...
```


Spójność systemu plików

- Dane systemu plików są zmieniane w pamięci, a dopiero później zmiany zapisywane są na dysku. W razie awarii system plików pozostaje w stanie niespójnym.
- Do przywracania spójności metadanych służy komenda `fsck`.
- **Dla starszych systemów plików (np. EXT3) sprawdzanie spójności systemu plików jest czynnością bardzo czasochłonną.**

Spójność systemu plików

Jak fsck sprawdza spójność plików?

1. dla każdego pliku (i-węzła) tworzony jest licznik
2. przeglądana jest lista plików (poczynając od korzenia systemu plików)
3. po napotkaniu pliku zwiększany jest licznik jego i-węzła
4. dla każdego i-węzła porównywany jest stan jego licznika z liczbą dowiązań zapisaną w i-węźle

Jeśli system jest spójny, to liczba dowiązań jest równa stanowi licznika.

Spójność systemu plików

Rodzaje niespójności plików:

- liczba dowiązań jest mniejsza niż stan licznika (liczba pozycji w katalogach) – niemożność odzyskania i-węzła
- liczba dowiązań jest większa niż stan licznika – zwolnienie bloków przy usunięciu któregoś z plików

Spójność systemu plików

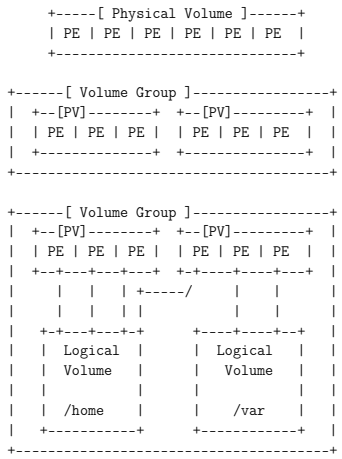
Systemy plików z kroniką, dziennikowe systemy plików (JFS, *Journaling File System*): EXT3|4, ReiserFS, XFS

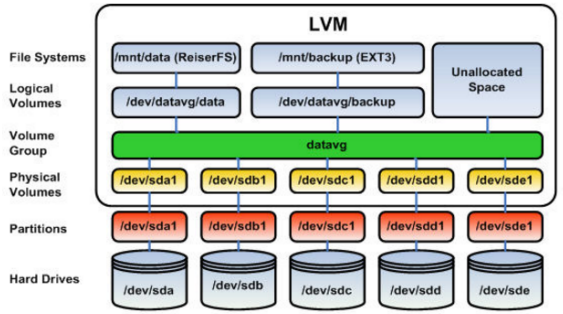
Zalety systemu plików EXT3|4:

- dostępność
- integralność danych – EXT3|4 zapewnia mocniejszą integralność danych w razie awarii systemu
- szybkość – pomimo pisania niektórych danych dwa razy (do pliku kroniki i do bloków systemu plików), EXT3|4 zapewnia zwiększoną przepustowość, gdyż optymalizuje ruch głowic

Zarządca woluminów logicznych (LVM, *Logical Volume Manager*)

- Fizyczna przestrzeń na dysku (partycje dyskowe) po uzupełnieniu o dane administracyjne jest określana jako wolumin fizyczny (PV, *Physical Volume*) składający się z fizycznych ekstentów (PE, *Physical Extent*).
- Grupę fizycznych ekstentów można przypisać do grupy woluminu (VG, *Volume Group*), na który mogą się składać fizyczne ekstenty pochodzące z tego samego lub różnych dysków.
- Fizyczne ekstenty z danej grupy woluminu mogą zostać przydzielone do logicznego woluminu (LV, *Logical Volume*), który jest odpowiednikiem tradycyjnej partycji.
- Na woluminie logicznym tworzony jest system plików.





za: http://linux.macrofinances.com/index.php/2015/12/29/logical-volume-management/

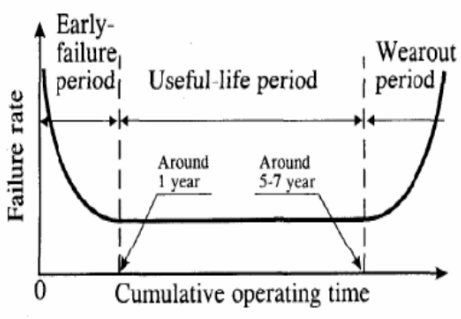
Zalety LVM

- dynamiczna zmiana wielkości systemu plików (`lvextend`, `lvreduce`, `resize2fs`).
- dodawanie przestrzeni fizycznej
- przesuwanie grupy woluminu na inny nośnik fizyczny
- tworzenie „zdjęć" systemu plików
- przeplot (*striping*) – zapisywanie danych związanych z danym woluminem logicznym na woluminach fizycznych, które go tworzą

Macierze dyskowe (RAID)

- Od 1985 do około 2000 r. nastąpił 100-krotny wzrost mocy procesorów, ale tylko 4-krotny wzrost wydajności dysków.
- W 1987 ukazała się praca Pattersona, Gibsona i Katza pt. *A Case for Redundant Arrays of Inexpensive Disks (RAID)* omawiająca rozmaite układy tanich dysków (RAID-1,. . .,RAID-5), które łaczone są w taki sposób, aby cały układ uzyskiwał wydajność lepszą niż pojedynczy, duży i drogi dysk *(Single Large Expensive Drive (SLED))*.
- RAID =
  Redundant Array of Inexpensive|Independent|Interchangable Disks
  nadmiarowa macierz tanich|niezależnych|zamiennych dysków
- RAID-0 = poziomu 0 nie zapewniająca nadmiarowości!
- Dla macierzy złożonej z N dysków średni czas międzyawaryjny (MTFB, *Mean Time Between Failures*) ulega skróceniu N razy.

Macierze dyskowe – MTBF[78]



[78] http://static.usenix.org/events/fast07/tech/schroeder/schroeder.pdf

Macierze dyskowe[79]

W ramach układu RAID wiele dysków tworzy jeden logiczny dysk dzięki podejściu zwanym **paskowaniem/przeplotem** (*striping*). Przestrzeń każdego z dysków jest dzielona na paski/człony (512 B, 1 K, 2 K, 4 K, ...), które są przydzielane rotacyjnie i przeplatane. Wielkość pasków zależy od stosowanych aplikacji i wielkości operacji wejścia/wyjścia (warstwa, to wielkość paska pomnożona przez liczbę dysków).

Nakładanie operacji wejścia/wyjścia dla różnych dysków powoduje nierównomierny rozkład danych i różne obciążenie dysków.

RAID poprzez stosowanie paskowania (i synchronicznie działających dysków) usuwa te niedogodności. Dane są równomiernie rozkładane na dyskach, a zapis/odczyt danych następuje równocześnie dla wszystkich dysków.

[79] Common RAID Disk Data Format Specification

RAID – poziomy

- **RAID-0** (*striping*) brak nadmiarowości, paskowanie zapewnia zwiększenie przepustowości, awaria któregoś z dysków oznacza utratę danych
- **RAID-1** (*mirroring*) dane są zapisywane równocześnie na dwóch (lub więcej) dyskach, większa szybkość czytanie (różne bloki mogą być równocześnie czytane z różnych dysków); doskonała ochrona danych w przypadku awarii pojedynczego dysku
- RAID-2 paskowanie na poziomie bitów z użyciem kodu Hamminga do korekcji błędów; poziom rzadko używany, bo dyski SCSI/IDE posiadają wbudowany system korekcji
- RAID-3 paskowanie na poziomie pojedynczego bajta z parzystością przechowywaną na wydzielonym dysku (wąskie gardło); zapis wolniejszy z uwagi na konieczność modyfikowania bajta parzystości (dobra prędkość przy zapisach sekwencyjnych); niezbyt wysoki koszt na jeden megabajt przestrzeni dyskowej

Parzystość w RAID-3

dysk	dane	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	137	1	0	0	0	0	1	1	1
2	11	0	0	0	0	1	0	1	1
XOR	137⊕11	1	0	0	0	1	1	0	0
3	96	0	1	1	0	0	0	0	0
XOR	(137⊕11)⊕96	1	1	1	0	1	1	0	0
4	157	1	0	0	1	1	1	0	1
XOR	(137⊕11)⊕96⊕157	0	1	1	1	0	0	0	1
suma		+	−	−	−	+	+	+	−

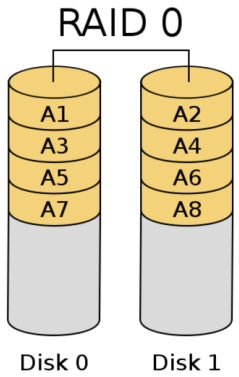
RAID – poziomy

- RAID-4 paskowanie na poziomie pojedynczego bloku z blokiem parzystości przechowywanym na wydzielonym dysku; szybkość odczytu jak dla RAID-0
- **RAID-5** RAID-4 z blokiem parzystości zapisywanym na różnych dyskach (przyspiesza to zapis małych plików); niższa szybkość odczytu z uwagi na konieczność opuszczania bloków z danymi o parzystości); dobre rozwiązanie pod względem dostępności danych i wydajności
- **RAID-6** paskowanie na poziomie bloku, zdublowany blok parzystości
- **RAID-XY** (*Nested RAID*) RAID-Y z dyskami skonfigurowanymi jako RAID-X

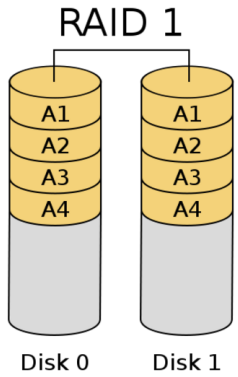
  Np. RAID-10: ochrona danych jak RAID-1 i zrównoleglenie zapisu jak w RAID-0; wysoki koszt przechowywania danych, ale bardzo dobra wydajność[80]

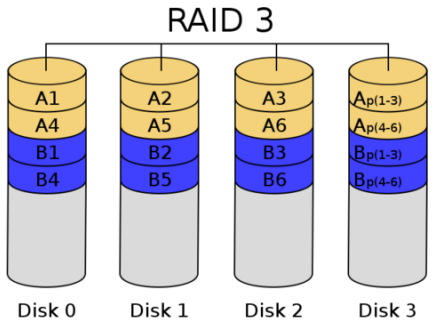
[80] Scot Lowe, *Non-standard RAID levels primer: RAID 1E*

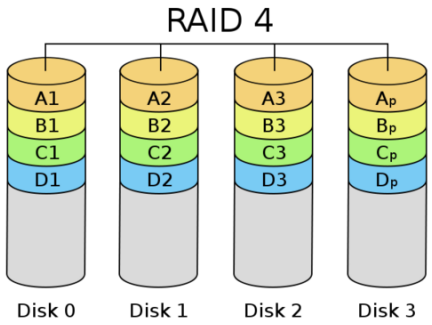
Poziomy RAID[81]

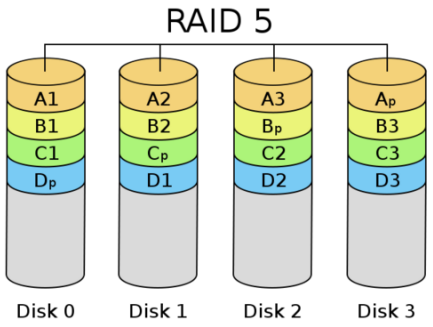


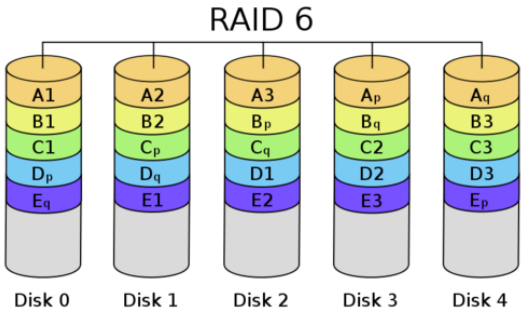
[81] http://www.linux-mag.com/id/7924/

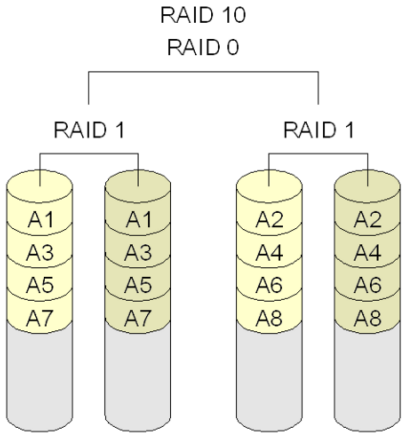





RAID 4
A1
B1
C1
D1
A2
B2
C2
D2
A3
B3
C3
D3
Ap
Bp
Cp
Dp
Disk 0
Disk 1
Disk 2
Disk 3


RAID 5
A1
B1
C1
Dp
A2
B2
Cp
D1
A3
Bp
C2
D2
Ap
B3
C3
D3
Disk 0
Disk 1
Disk 2
Disk 3


RAID 6
A1
B1
C1
Dp
Eq
A2
B2
Cp
Dq
E1
A3
Bp
Cq
D1
E2
Ap
Bq
C2
D2
E3
Aq
B3
C3
D3
Ep
Disk 0
Disk 1
Disk 2
Disk 3
Disk 4


RAID 10
RAID 0
RAID 1
RAID 1
A1
A3
A5
A7
A1
A3
A5
A7
A2
A4
A6
A8
A2
A4
A6
A8

RAID sprzętowy i programowy

RAID programowy pozwala wykorzystać różne dyski w dowolnej lokalizacji. Programowe realizacje macierzy dyskowych są wolniejsze niż sprzętowe.

RAID sprzętowy zmiejsza obciążenie procesora, zastosowanie pamięci nieulotnej do buforowania logu zapisów przyspiesza operacje zapisu, wewnętrzne optymalizacje zapisów/odczytów (dotyczy głównie RAID 5, wymagane 1-2 GB NVRAM).

Porównanie poziomów RAID[82]

Poziom RAID	Dostępność danych (awaria)	Wydajność odczytu	zapisu	odtwarzania
0	żadna	bdobra	bdobra	nie dotyczy
1	wspaniała	bdobra	dobra	dobra
5,6	dobra	dobra	znośna	słaba
10	wspaniała	bdobra	dobra	dobra
50,60	wspaniała	bdobra	znośna	znośna

[82] Advantages and Disadvantages of RAID Levels

Struktura systemów operacyjnych

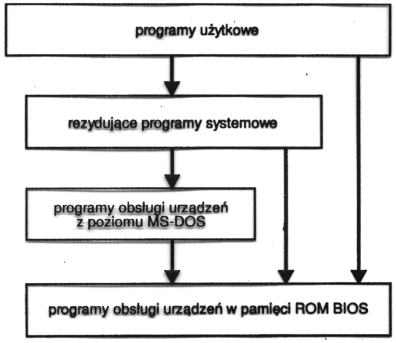
- jednolita (monolityczna) – zbiór procedur, które mogą się wzajemnie wywoływać; możliwość efektywnego wykorzystywania zasobów i otrzymania efektywnego kodu, podatność na błędy, trudność rozwijania i konserwowania systemu (współczesne systemy oferują ładowanie modułów na żądanie)
- warstwowa – moduły podzielowne w warstwy: każda warstwa spełnia funkcje, które zależą od warstwy bezpośrednio niższej (warstwa n+1-a odwołuje się do n-tej (czasami n-k-tej; możliwość odwołań w przeciwnym kierunku)

Struktura systemów operacyjnych

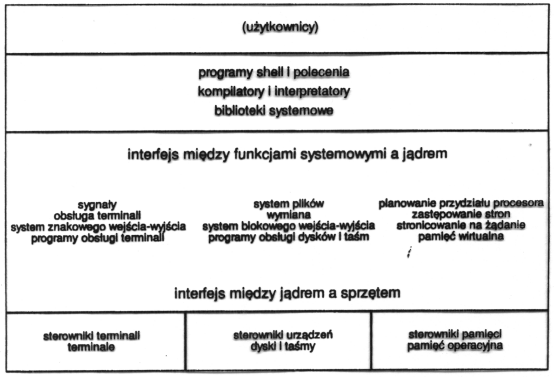
System warstwowy THE
(zaprojektowany i wykonany przez E.W.Dijkstrę i studentów w 1968)

5	The operator
4	User program
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Procesor allocation and multiprograming

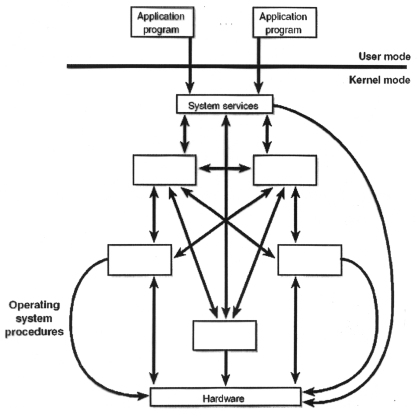
Struktura systemu MS-DOS



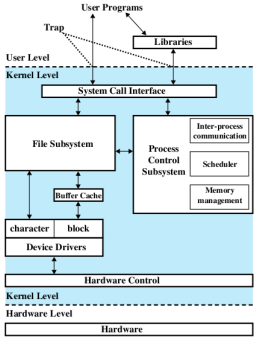
Struktura systemu Unix



Struktura systemu Unix

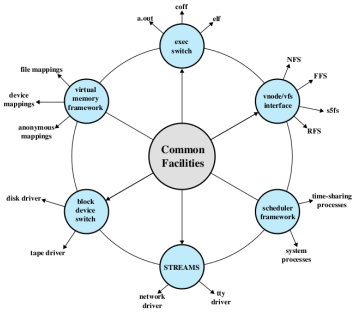


Struktura systemu Unix[83]



[83] Maurice J. Bach *The Design of the UNIX Operating System*

Struktura systemu Unix[84]

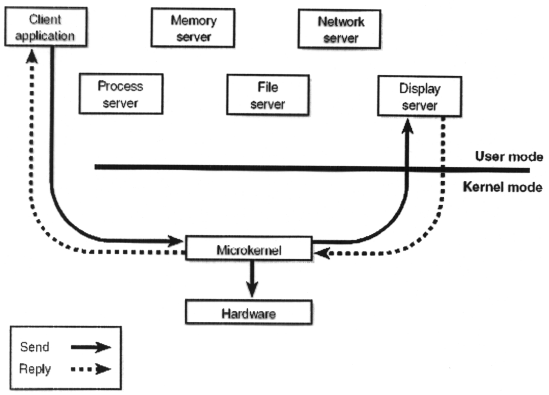


[84] Uresh Vahalia *UNIX Internals*

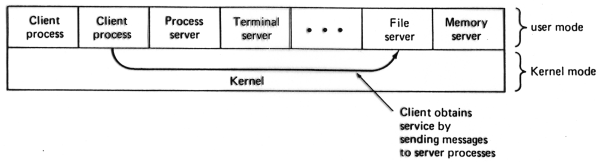
Struktura systemów operacyjnych

- klient-serwer – moduły spełniają wyraźnie określone, odrębne zadania
  - moduły nie są rozmieszczone w warstwach, ale są równorzędne
  - komunikacja odbywa się poprzez przesyłanie komunikatów, a nie wywoływanie procedur
  - mikrojądro (*microkernel*) – program centralny umożliwiający przekazywanie komunikatów i dostarczający innych udogodnień
  - przesuwanie wszystkiego „co się da" z przestrzeni jądra do przestrzeni użytkownika
  - łatwość rozszerzania mikrojądra i portowania na nowe architektury
  - zwiększona odporność na awarie i bezpieczeństwo (mniej kodu pracuje w trybie jądra)
- hybrydowe – modyfikowane mikrojądro w celu poprawy wydajności (Windows 2000, XP, Mac OS X)

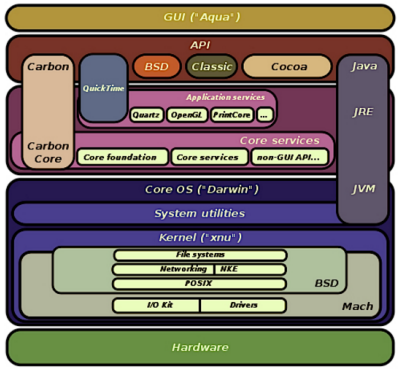
Struktura systemu klient-serwer



Struktura systemu klient-serwer



Struktura systemu hybrydowego[85]



[85] Architecture of OS X

Struktura systemów operacyjnych

- maszyna wirtualna
  - oddzielenie wieloprogramowości i maszyny rozszerzonej (wirtualnej); *virtual machine monitor* (*virtual monitor system*) ma bezpośredni kontakt ze sprzętem, realizuje wieloprogramowość i dostarcza wielu wirtualnych maszyn, które są kopiami wyjściowej maszyny
  - przykłady: CP/CMS (*Control Program/Cambridge Monitor System*), VM/370 – wersja komercyjna

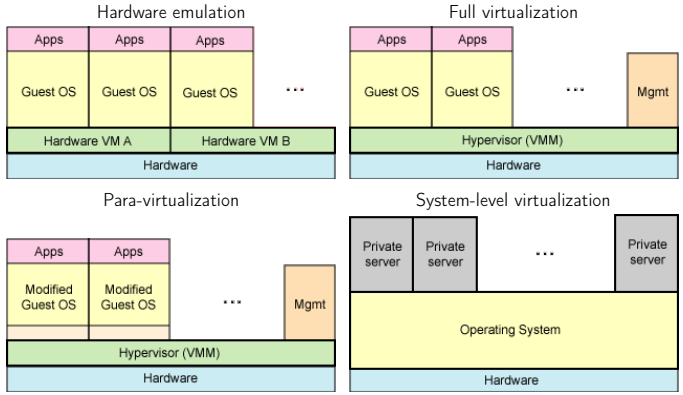
Struktura systemów operacyjnych: wirtualizacja

Rodzaje wirtualizacji:[86]

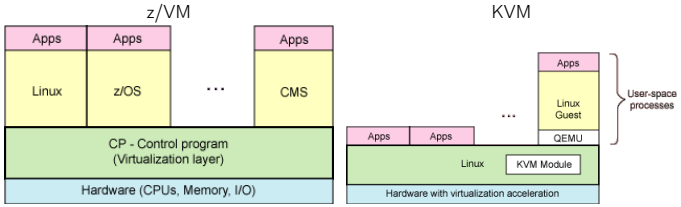
- emulacja sprzętu – wirtualna maszyna emuluje sprzęt inny niż sama wykorzystuje do wykonywania instrukcji (Bochs, QEMU)
- pełna wirtualizacja – udostępnianie sprzętu na potrzeby maszyny wirtualnej pracującej pod nadzorem dowolnego systemu operacyjnego, który musi wspierać wykorzystywany sprzęt (VMware, Microsoft Virtual Server, z/VM, Kernel Virtual Machine)
- parawirtualizacja – pełna wirtualizacja wspierana przez modyfikacje gościnnego systemu operacyjnego (Xen, User-mode Linux)
- wirtualizacja na poziomie systemu operacyjnego (OpenVZ, Linux Containers (LXC), Linux-VServer, Solaris Zones, FreeBSD Jails)
- wirtualizacja na poziomie biblioteki, np. Wine (częściowe Win32 API dla Linuksa), LxRun (częściowe API dla Solarisa)

[86] Emulation or Virtualization

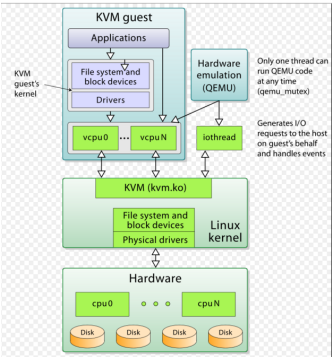
Rodzaje wirtualizacji



Przykłady wirtualizacji

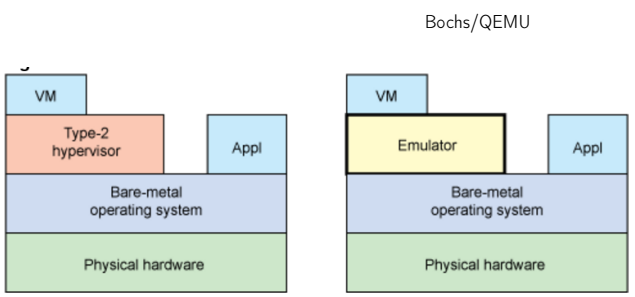


Przykłady wirtualizacji[87]



[87] Kernel-based Virtual Machine

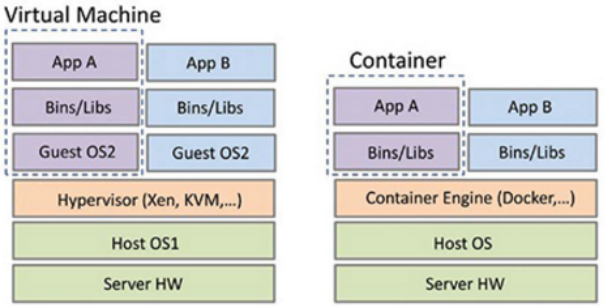
Przykłady wirtualizacji[88]



QEMU może działać jako emulator albo wirtualizator (Xen, KVM)

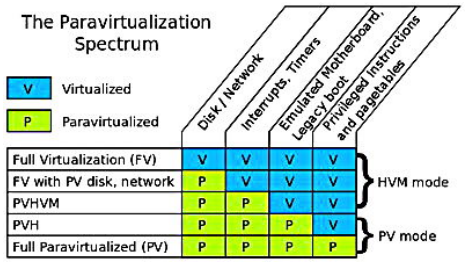
[88] I-bochs

Przykłady wirtualizacji[89]



[89] Understanding virtualization and containers

Zakresy parawirtualizacji[90]



[90] An Introduction to Full Virtualization With Xen (Part 1), Intro to the Paravirtualization Spectrum With Xen (Part 2)

Słownik skrótów

3DNow *3D NO Waiting* bez stanów oczekiwania 3D
AC *accumulator* akumulator
AGP *Accelerated Graphics Port* przyspieszony port graficzny, port akcelerowanej grafiki
AGU *Address Generation Unit* jednostka generowania adresów
ALU *Arithmetic-Logic Unit* jednostka arytmetyczno-logiczna
API *Application Programming Interface* interfejs programistyczny aplikacji
APIC *Advanced Programmable Interrupt Controller* zaawansowany programowalny kontroler przerwań
ATA *AT Attachement* końcówka AT
ASCII *American Standard Code for Information Interchange* amerykański standard kodowania na potrzeby wymiany informacji
BIOS *Basic Input/Output System* podstawowy system wejścia/wyjścia
BSD *Berkeley Software Distribution* wersja systemu uniksopodobnego z Berkeley
Btrfs *B-tree File System)* system plików oparty o B-drzewo

CD-ROM *Compact Disk Read-Only Memory* pamięć tylko do odczytu na płycie kompaktowej
CentOS *Community ENTerprise Operating System*
CHS *Cylinder/Head/Sector* cylinder/głowica/sektor
CFS *Completely Fair Scheduler* planista całkowicie sprawiedliwy
CIFS *Common Internet File System* wspólny internetowy system plików
CISC *Complex Instruction Set Computer* komputer o pełnej liście rozkazów
CLI *Command Line Interface* wierszowy interfejs komend
CMOS *Complementary Metal-Oxide Semiconductor* komplementarny półprzewodnik tlenkowy
COW *Copy On Write* kopiowanie przy zapisie
CPU *Central Processing Unit* jednostka centralna
CRC *cyclic redundancy check* cykliczna kontrola nadmiarowa
DAC *discretionary access control* uznaniowa kontrola dostępu
DMA *Direct Memory Access* bezpośredni dostęp do pamięci
DMI *Direct Media Interface* magistrala łącząca mostek północny z południowym

DRAM *Dynamic RAM* dynamiczna pamięć RAM
EDVAC *Electronic Discrete Variable Computer*
EEPROM *Electrically Erasable and Programmable ROM* wymazywalna i programowalna pamięć stała
EIDE *Extended Integrated Device Electronics* scalona elektronika obsługi dysku
EISA *Extended Industry Standard Architecture* rozszerzona standardowa architektura przemysłowa
EMS *Expanded Memory System* układ pamięci rozszerzonej
ENIAC *Electronic Numerical Integrator and Computer* elektroniczne urządzenie numeryczne całkujące i liczące
EPROM *Erasable and Programmable ROM* wymazywalna i programowalna pamięć stała
EXT2 *Second Extended File System* drugi rozszerzony system plików
EXT3 *Third Extended File System* trzeci rozszerzony system plików
EXT4 *Fourth Extended File System* czwarty rozszerzony system plików
FAT *File Allocation Table* tablica alokacji plików

FDD *Floppy Disk Drive* stacja dyskietek
FLOPS *Floating-point operations per second* liczba operacji zmiennopozycyjnych na sekundę
FIFO *First In, First Out* pierwszy nadszedł, pierwszy obsłużony
FSB *Front System Bus* przednia magistrala systemowa
FSF *Free Software Fundation* Fundacja Wolnego Oprogramowania
GDT *Global Descriptor Table* globalna tablica deskryptorów
GNU *Gnu is Not Unix* Gnu nie jest Uniksem
GPL *General Public Licence* Ogólna Licencja Publiczna
GPU *Graphics Processing Unit/Graphical Processor Unit* procesor graficzny
GUI *Graphical User Interface* graficzny interfejs użytkownika
HDD *Hard Disk Drive* dysk twardy
HT *Hyper-Threading* hiperwątkowość
HVD *High Voltage Differential* wysokie napięcie różnicowe
IBR *instruction buffer register* rejestr bufora rozkazów
IEU *Integer Execution Unit* jednostka wykonawcza dla liczb stałopozycyjnych

I/O *Input/Output* wejście-wyjście
IPC *InterProcess Communication* komunikacja międzyprocesowa
IR *instruction register* rejestr rozkazów
IRQ *Interrupt ReQuest* linia/żądanie przerwania
ISA *Instruction Set Architecture* architektura zbioru/zestawu instrukcji (model programowy procesora)
ISR *Interrupt Service Routine* procedura obsługi przerwania
JCL *Job Control Language* język kontroli zadań
JFS *Journaling File System* system plików z kroniką
KVM *Kernel-based Virtual Machine* maszyna wirtualna realizowana przez jądro systemu operacyjnego
LBA *Linear Block Addressing* liniowa adresacja bloków
LDT *Local Descriptor Table* lokalna tablica deskryptorów
LSI *Large Scale Integration* duży stopień scalenia
LVM *Logical Volume Manager* menadżer woluminów logicznych
MAC *mandatory access control* obowiązkowa kontrola dostępu
MAR *memory address register* rejestr adresowy pamięci

MBR *memory buffer register* rejestr bufora pamięci
MBR *Master Boot Record* główny rekord rozruchowy, pierwszy sektor dysku
MQ *multiplier-quotier* rejestr mnożenia i dzielenia
MIMD *Multiple Instruction Multiple Data* wiele instrukcji wiele danych
MMX *MultiMedia Extensions* rozszerzenia multimedialne
MPP *Massively Parallel Processing* przetwarzanie masywnie równoległe
MS-DOS *Microsoft Disk Operating System* system operacyjny dla komputerów zgodnych z IBM PC
MTA *Mail Transport Agent* agent transportu poczty
MTBF *Mean Time Between Failures* średni czas międzyawaryjny
NMI *NonMaskable Interrupts* przerwania niemaskowalne
NVRAM *Non-Violatile RAM* nieulotna pamięć RAM
NTFS *New Technology File System* system plików nowej technologii
NUMA *Non-Uniform Memory Access* niejednakowy/niejednolity dostęp do pamięci
NVRAM *Non-Volatile RAM* nieulotna pamieć RAM
PATA *Parallel AT Attachement* równoległa końcówka AT

PC *program counter* licznik programu
PC *personal computer* komputer osobisty
PCI *Peripheral Component Interconnect* interfejs komponentów peryferyjnych
PnP *Plug and Play* podłącz i używaj
POSIX *Portable Operating System Interface for UNIX* przenośny interfejs systemu operacyjnego dla Uniksa
PROM *Programmable ROM* programowalna pamięć stała
RAID *Redundant Array of Inexpensive Disks* nadmiarowa tablica (macierz) niedrogich dysków
RAM *Random Access Memory* pamięć swobodnego dostępu
RHEL *Red Hat Enterprise Linux*
RISC *Reduced Instruction Set Computer* komputer o zredukowanej liście rozkazów
ROM *Read-Only Memory* pamięć stała/tylko do odczytu
RPC *Remote Procedure Call* zdalne wywołanie procedury
SATA *Serial AT Attachement* szeregowa końcówka AT

SAS *Serial Attached SCSI* szeregowy interfejs SCSI
SCSI *Small Computer System Interface* interfejs małych systemów komputerowych
SIMD *Single Instruction Multiple Data* pojedyncza instrukcja wiele danych
SJF *Shortest Job First* najkrótsze zadanie najpierw
SMB *Server Message Block* blok komunikatów serwera
SMP *Symmetric MultiProcessing* symetryczne przetwarzanie na wielu procesorach
SPOOL *Simultaneous Peripheral Operation On-Line* jednoczesna bezpośrednia praca urządzeń
SRAM *Static RAM* statyczna pamięć RAM
SSD *Solid State Device* dysk/napęd półrzewodnikowy/urządzenie półprzewodnikowe
SSE *Streaming SIMD Extensions* rozszerzenia strumieniowe SIMD
TLB *Translation Lookaside Buffers* bufory translacji adresów
TSL *Test and set Lock* testuj i załóż rygiel
TSS *Task Status Segment* segment statusu zadania

USB *Universal Serial Bus* uniwersalna szyna szeregowa
VFS *Virtual File System* wirtualny system plików
VLSI *Very Large Scale Integration* bardzo duża skala integracji