System Linux

1.1 Historia

Prace nad systemem zapoczątkował Linus Thorvald. Wywodzi się z wczesniejszego systemu MINIX. Pierwsza wersja powstała w 1991 na procesor Intel 386. Początkowo Linux liczył tylko 9300 linii kodu w języku C i 950 linii kodu assemblera.

1970	Powstał system operacyjny Unix opracowany przez Koen Thompson i Denis Ritchie z Bell Laboratories. Był to początkowo system otwarty. Z czasem na skutek ograniczeń licencyjnych kod stawał się coraz bardziej zamkniety.
1982	Richard Stallman zapoczątkował projekt GNU (ang. *Gnu is Not Unix*) wprowadzając koncepcję oprogramowania otwartego
1987	Andrew Tannenbaum opublikował w książce *Operating System Design and Implementation* projekt i wersję źródłową systemu MINIX przeznaczonego dla procesorów 386. Kod był otwarty ale rozpowszechnianie ograniczone.
1991	Linus Torvald opublikował projekt który stał się zaczątkiem jądra Linuksa.
1992	Opublikowano Linuksa jako część wolnego oprogramowania GNU
1994	Ukazuje się wersja jądra 1.0 Wszystkie podstawowe moduły jądra zostały ukończone. Wersja ta liczy 165 tysięcy linii kodu.
1996	Wizerunek pingwina stał się oficjalną maskotką Linuksa
1996	Ukazała się wersja jądra 2.0 obejmująca między innymi przetwarzanie wieloprocesorowe SMP. Wersja ta liczy 450 tysięcy linii kodu i 8000 linii kodu assemblera.
2003	Ukazała się wersja jądra 2.6 zawierająca wiele zaawansowanych właściwości
2011	Ukazała się wersja jądra 3.0

Niezwykłą cechą Linuxa jest sposób jego udostępniania. Jest on dostępny za darmo na warunkach licencji GPL (ang. *GNU Public License*) opracowanej przez Richarda Stallmana założyciela fundacji Free Software Foundation.

Oprogramowanie GNU może być używane do celów tak nie komercyjnych jak i komercyjnych. Warunkiem jest publikowania oprogramowania wraz z jego kodem źródłowym.

1.2 Własności Linuxa

System operacyjny jest zbiorem programów sterujących pracą komputera.

- Oddziela użytkownika od złożonego sprzętu i tworzy środowisko w którym wykonują się programy.
- Dostarcza interfejsu do komunikacji z użytkownikiem.

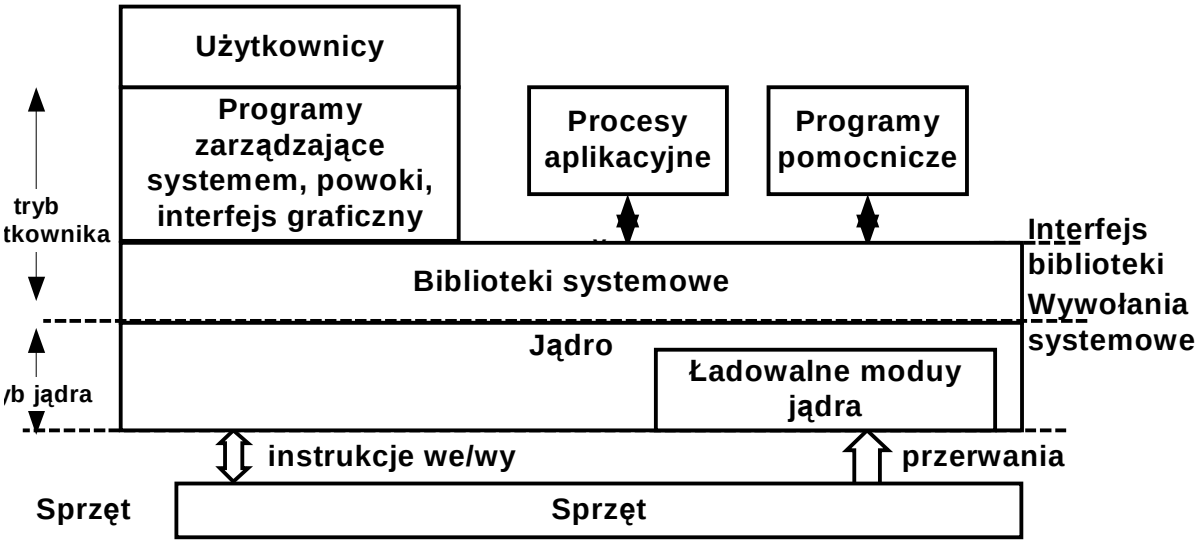
Najważniejsze cechy systemu Linux są następujące:

- Implementuje abstrakcję procesów.
- Jest w stanie obsługiwać wielu użytkowników. Zaimplementowane mechanizmy kontroli dostępu do zasobów pozwala na bezpieczne ich współistnienie.
- Implementuje abstrakcję plików. Plikami są tak pliki regularne, katalogi jak i urządzenia. Umożliwiają dostęp do bardzo nieraz różnych zasobów w jednolity sposób.
- Implementuje pamięć wirtualną. Pamięć podzielona jest na strony a dostęp do stron jest jednolity niezależnie czy są one umieszczone w pamięci operacyjnej czy zewnętrznej.
- Implementuje zaawansowane mechanizmy ochrony zasobów oparte na segmentacji pamięci i dwóch trybach pracy procesora - trybie systemowym (ang. system) i użytkownika (ang. user). Pamięć używana przez jeden proces jest chroniona przed innymi procesami. Umożliwia to implementację niezawodnie działającego oprogramowania.
- Implementuje stos protokołów TCP/IP. Zapewnia to możliwość budowy systemów rozproszonych i dostęp do sieci Internet.
- Umożliwia wykorzystanie wielu procesorów zapewniając przetwarzanie równoległe. Obsługiwany jest model SMP (ang. *Symetric Multiprocessing*).

1.3 Składowe systemu

System składa się z trzech głównych elementów

- Jądro – odpowiada za realizację wszystkich istotnych abstrakcji systemu: pamięci witrualnej, procesów i plików.
- Biblioteki systemowe – określają standardowy zestaw wywołań systemowych za pomocą których aplikacje mogą współdziałać z jądrem.
- Programy i demony systemowe – programy wykonujące funkcje systemowe i pomocnicze. W tym demony systemowe – programy pracujące w sposób ciągły i realizujące różne funkcje (ftp, demon sieciowy, itd.).



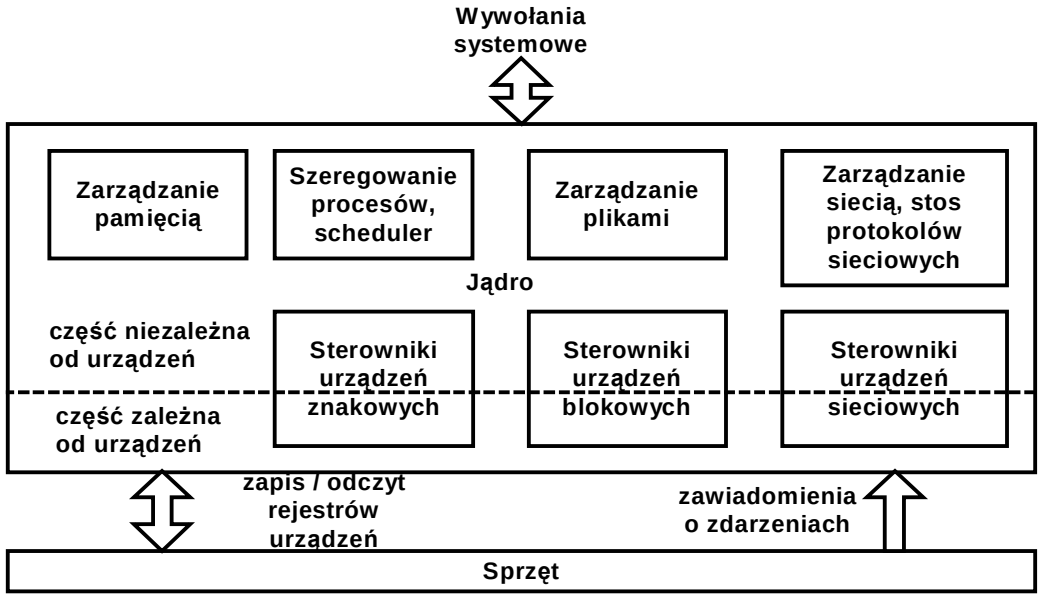
Rys. 0-1 Ogólny schemat systemu Linux

1.4 Jądro

Jądro dostarcza wszystkich podstawowych funkcji do działania komputera. Należą do nich następujące funkcje:

- Zarządzania pamięcią
- Tworzenie wątków i procesów
- Szeregowanie procesów
- Obsługa mechanizmów komunikacji międzyprocesowej IPC (ang. *Inter Process Communication*)
- Obsługa systemu wejścia wyjścia
- Obsługa systemu plików
- Obsługa sieci

Interfejs do usług jądra realizowany jest poprzez biblioteki systemowe. Dostarczają one bardziej złożonych wersji podstawowych usług systemowych.



Rys. 0-1 Koncepcyjny schemat jądra Linuksa

Dodatkowo jądro Linuksa posiada takie własności jak:

- Przenośność – dostępne jest dla wielu architektur sprzętowych
- Skalowalność – może być wykonywane zarówno na mikro sterownikach jak i na superkomputerach
- Wysoki poziom bezpieczeństwa – ze względu na rozpowszechnienie zostało dobrze przetestowane
- Zgodność ze standardami – np. POSIX 1003.1
- Modularność
- Rozszerzalność – dostępny kod źródłowy i wiele przykładów jak jądro rozbudowywać.

1.4.1 Tryb jądra i tryb użytkownika

Cały kod jądra i używane przez ten kod struktury danych utrzymywane są w jednej przestrzeni adresowej.

- Kod jądra wykonywany jest w uprzywilejowanym trybie procesora (ang. *Privileged Mode*),
- procesy wykonywane są w trybie użytkownika (ang. *User Mode*).

W trybie użytkownika pewne potencjalnie niebezpieczne instrukcje nie mogą być wykonywane. Do potencjalnie niebezpiecznych instrukcji zaliczamy:

- Instrukcje wejścia / wyjścia, dokonujące manipulacji na urządzeniach zewnętrznych
- Instrukcje dostępu do konfiguracyjnych i sterujących rejestrów procesora
- Instrukcje zmiany niektórych flag procesora
- Instrukcje zatrzymywania procesora

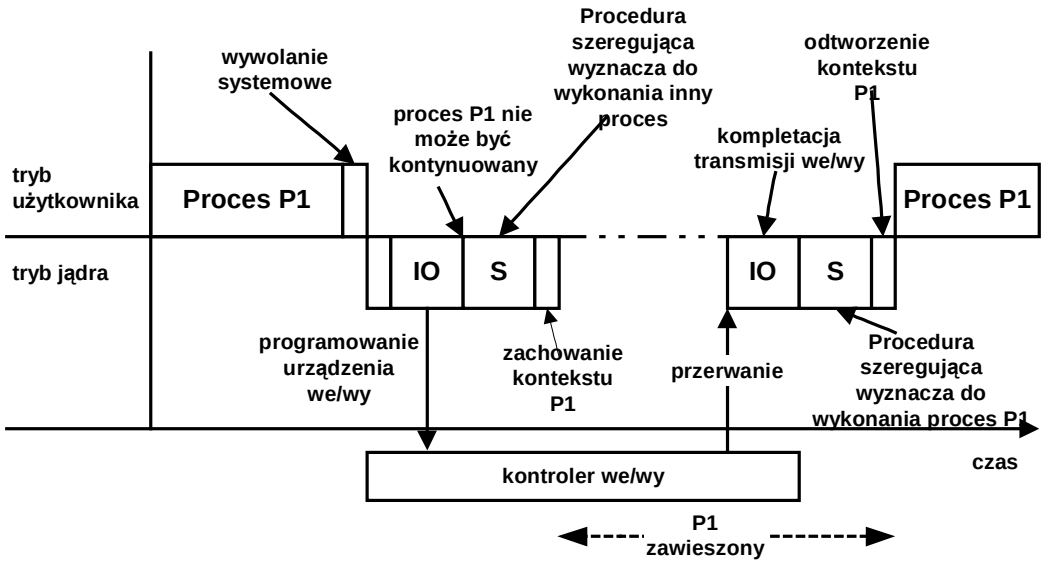
Próba wykonania niebezpiecznych instrukcji w trybie użytkownika powoduje powstanie wyjątku (ang. *Exception*) i w konsekwencji zakończenie procesu.

Gdy proces potrzebuje wykonania akcji której nie może sam przeprowadzić , na przykład kontaktu z urządzeniem zewnętrznym, formułuje odpowiednie zlecenie w postaci wywołania systemowego (ang. *System Call*) i przekazuje te zlecenie do jądra. Następuje przełączenie procesu w tryb jądra (ang. *Kernel Mode*).

Przykład:
read(int fh, void * bufor, int size).

Jądro może zapoczątkować żądaną przez proces akcję, np. zainicjować transmisję z/do urządzenia wejścia/wyjścia. Do czasu jej zakończenia proces nie może być kontynuowany. Dlatego też sterowanie może nie powrócić do wykonującego wywołanie systemowe procesu, gdyż ten i tak nie może być kontynuowany. Zamiast tego wykonywana jest procedura szeregująca procesy w wyniku której wyznaczany jest kolejny proces który może być kontynuowany.

Po jakimś czasie żądane przez proces zasoby staną się dostępne, co zostanie zasygnalizowane przerwaniem od kontrolera urządzenia wejścia/wyjścia. Wtedy procedura szeregująca może ponownie przekazać sterowanie do zawieszonego procesu i będzie on w trybie użytkownika kontynuowany.



Rys. 0-2 Realizacja wywołania systemowego, przejście procesu P1 pomiędzy trybami jądra i użytkownika

1.4.2 Moduły jądra

Cały kod jądra i używane przez ten kod struktury danych utrzymywane są w jednej przestrzeni adresowej.

Jądro umożliwia dynamiczne dodawać i usuwać fragmenty kodu nazywane modułami. Potrzeba ta wynika z różnorodności sprzętu na którym system może być wykonywany. Standardowe jądro zawiera sterowniki typowych urządzeń. Po załadowaniu może być ono uzupełnione o moduły sterowników urządzeń występujących w danym komputerze.
Moduły muszą być wcześniej skompilowane. Nie trzeba za to rekompilować całego jądra.

Administrowanie modułami zawiera składowe:

- Zarządzanie modułami – umożliwia ładowanie modułów do pamięci i umożliwienie ich integracji z jądrem
- Rejestracja modułów sterujących – pozwala informować jądro o dostępności sterownika nowego urządzenia.
- Mechanizm rozwiązywania konfliktów – umożliwia modułom rezerwację zasobów sprzętowych aby nie doszło do konfliktu w dostępie wielu sterowników do jednego urządzenia.

Aktualnie załadowane moduły jądra można uzyskać wyświetlając zawartość pliku: **`/proc/modules`**

Sformatowany wydruk można także uzyskać za pomocą polecenia **`modinfo`.**

```
$modinfo
Module                  Size  Used by
pl2303                 11812  0
ext4                  256936  0
vfat                    6570  0
fat                    34912  1 vfat
usb_storage            31033  0
usbserial              22100  2 pl2303,ftdi_sio
l2cap                  21721  3 bnep
...
```

Ekran 0-1 Uzyskanie listy załadowanych modułów za pomocą polecenia lsmod (fragment)

1.4.3 Pliki jądra

Jądro znajduje się w katalogu /boot w postaci skompresowanego pliku o nazwie **`vmlinuz-wersja`**.

```
-rw-r--r-- 1 root root  111135 2012-09-23  config-2.6.32-5-686
drwxr-xr-x 3 root root    4096 02-02 11:25 grub
-rw-r--r-- 1 root root 8707112 02-01 14:29 initrd.img-2.6.32-5-686
-rw-r--r-- 1 root root 1295399 2012-09-23  System.map-2.6.32-5-686
-rw-r--r-- 1 root root 2301536 2012-09-23  vmlinuz-2.6.32-5-686
```

Ekran 1 Zawartość katalogu /boot systemu Linux Debian squeeze

`System.map-2.6.32-5-686` - informacje o położeniu obiektów jądra (zmiennych i funkcji)

`initrd.img-2.6.32-5-686` - plik zawiera skompresowany początkowy system plików umieszczany w pamięci operacyjnej. Plik ten zawiera narzędzia niezbędne do zamontowania głównego systemu plików (ang. *root filesystem*).

`config-2.6.32-5-686` - plik jest plikiem konfiguracyjnym zawierającym informacje o konfiguracji bieżącego jądra.

`grub` - katalog który zawiera program ładujący jądro i jego pliki pomocnicze.

1.5 Zarządzanie procesami

Proces jest podstawowym mechanizmem umożliwiającym wykonanie jakichkolwiek czynności.

System dostarcza dwóch podstawowych czynności do zarządzania procesami:

- Utworzenie procesu – funkcja **`fork`**.
- Wykonanie nowego programu – funkcja **`execve`**.

Całość informacji niezbędnych do administrowania procesem dzieli się na trzy grupy:

- Tożsamość procesu
- Środowisko procesu
- Kontekst procesu

Tożsamość procesu:

- Identyfikator procesu – PID procesu, grupa procesów
- Uwierzytelnienia – identyfikator użytkownika i grupy
- Indywidualność (ang. *personality*) – z cechy tej korzystają biblioteki emulacji

Środowisko procesu

Na środowisko procesu składają się dwa wektory:

- Wektor argumentów – wykaz argumentów wywołania programu
- Wektor środowiska – zestaw par NAZWA=WARTOŚĆ

Środowisko procesu dziedziczone jest z procesu macierzystego.

Kontekst procesu

Kontekst procesu zawiera informacje:

- Kontekst planowania: rejestry, priorytet bieżący, nie obsłużone sygnały
- Informacje rozliczeniowe: skumulowany czas procesora
- Tablica plików: zawiera wskaźniki do otwartych plików utrzymywanej przez jądro.
- Tablica obsługi sygnałów: tablica specyfikująca akcje mające być wykonane przy nadejściu sygnałów

- Kontekst pamięci wirtualnej: opisuje zawartość przestrzeni adresowej procesu

System plików /proc

Informacja dotycząca różnych aspektów działania jądra, w tym wykonywanych procesów, dostępna jest dla użytkownika w postaci wirtualnego systemu plików **/proc**.

```
$ls /proc
1           10          1080        11          1122        11592
11          8612        13          1394        14          4
…
asound      buddyinfo   bus         cgroups     cmdline     cpuinfo
crypto      devices     diskstats   dma         driver      execdomains
fb          filesystems fs          interrupts  iomem       ioports
irq         kallsyms    kcore       keys        key-users   kmsg
kpagecount  kpageflags  loadavg     locks       meminfo     misc
modules     mounts      mpt         mtrr        net         pagetypeinfo
partitions  sched_debug scsi        self        slabinfo    softirqs
stat        swaps       sys         sysvipc     timer_list  timer_stats
tty         uptime      version     vmallocinfo vmstat      zoneinfo
```

Ekran 2 Zawartość katalogu /proc w systemie Linux – Debian squeeze

Występujące w katalogu **/proc** podkatalogi będące cyframi, reprezentują procesy systemu. Informację o dowolnym procesie zawarte są w reprezentującym ja podkatalogu.

```
$ps
  PID TTY          TIME CMD
 4835 pts/2    00:00:01 bash
 4959 pts/2    00:00:00 ps
ls /proc/4835
attr       auxv       cgroup     clear_refs     cmdline
cpuset     cwd        envi       Exe            fd
fdinfo     io         limits     login          maps
mem        mountinfo  mounts     mountstats     net
pagemap    personality           root           sched
essionid   smaps      stack      stat           statm
status     syscall    task
```

Ekran 3 Informacja o procesie zawarta w katalogu proc

Uzyskane wyniki pokazano we fragmencie poniżej.

```
$cat /proc/4835/stat
Name:     bash State:    S (sleeping)   Tgid:     4835
Pid:4835  PPid:     2661
FDSize:   256       Groups:   24 25 27 29 30 44 46 108 109 115
1000
VmPeak: 6560 kB     VmSize:  6496 kB         VmLck:           0
kB
…
Threads:  1
```

Ekran 4 Informacja o statusie procesu zawarta w pliku stat

1.6 Programy trybu użytkownika

System Linux zawiera wiele programów działających w trybie użytkownika. Są to:

- narzędzia niezbędne do rozpoczęcia pracy systemu,
- konfigurowania urządzeń,
- ładowania modułów jądra
- także serwery systemowe.

Serwery systemowe obsługują informacje napływające z sieci, inicjują pracę użytkowników itd.

W skład systemu wchodzą też programy służące do obsługi systemu plików, graficznego interfejsu użytkownika, kompilatory i inne narzędzia niezbędne do codziennej pracy użytkownika.

1.7 System plików

1.7.1 Pliki i atrybuty

Plik jest podstawową abstrakcją używaną w systemach operacyjnych. Pozwala na traktowanie dużego zbioru zasobów w jednolity sposób.

W systemie Linux prawie wszystkie zasoby są plikami.
Dane i urządzenia są reprezentowane przez abstrakcję plików.

Mechanizm plików pozwala na jednolity dostęp do zasobów tak lokalnych jak i zdalnych za pomocą poleceń i programów usługowych wydawanych z terminala, uruchamianych programów i interfejsu graficznego.

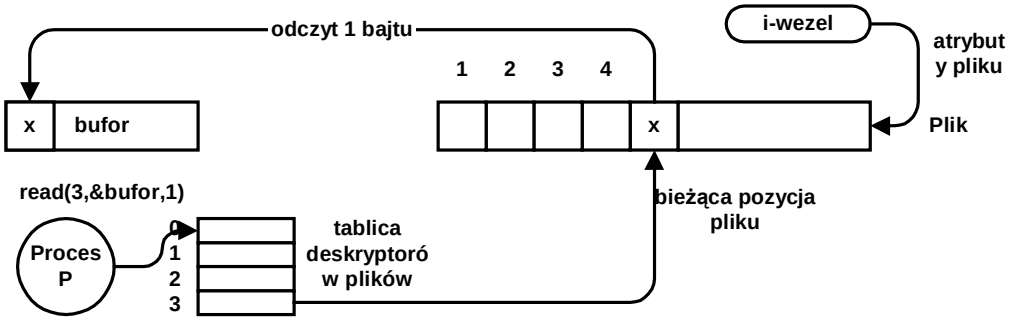
Plik jest obiektem abstrakcyjnym z którego można czytać i do którego można pisać.
Z plikiem związane są jego atrybuty (wielkość, prawa dostępu, czas dostępu, położenie,...) zapisane w tak zwanym i-węźle który położony jest w bloku organizacyjnym woluminu.

Typ pliku	Plik regularny, katalog, gniazdko, kolejka FIFO, urządzenie blokowe, urządzenie znakowe, link.
Prawa dostępu	Prawo odczytu, zapisu, wykonania określone dla właściciela pliku, grupy do której on należy i innych użytkowników systemu
Wielkość	Wielkość pliku w bajtach
Właściciel pliku	UID właściciela pliku
Grupa do której należy właściciel	GID grupy do której należy właściciel pliku
Czas ostatniej modyfikacji	Czas kiedy nastąpił zapis do pliku
Czas ostatniego dostępu	Czas kiedy nastąpił odczyt lub zapis do pliku
Czas ostatniej modyfikacji statusu	Kiedy zmieniano atrybuty takie jak prawa dostępu, właściciel, itp
Liczba dowiązań	Pod iloma nazwami (ang. *links*) występuję dany plik
Identyfikator urządzenia	Identyfikator urządzenia na którym plik jest pamiętany

Tabela 1 Atrybuty pliku

Gdy proces P chce uzyskać dostęp do pliku musi go otworzyć wykonując funkcję:
`fh = open(ścieżka_dostępu, flagi)`

Dostęp do pliku odbywa się za pomocą systemowych funkcji **`read`** i **`write`** i jest sekwencyjny.



Rys. 0-3 Ilustracja dostępu do pliku

Oprócz plików regularnych i katalogów w systemie plików obecne są pliki specjalne. Zaliczamy do nich łącza symboliczne, kolejki FIFO, gniazda, urządzenia blokowe.

Symbol	Znaczenie
`-`	Zwykły plik (regularny)
`d`	Katalog
`l`	Dowiązanie symboliczne (link symboliczny)
`c`	Urządzenie znakowe
`b`	Urządzenie blokowe
`p`	Łącze nazwane (kolejka FIFO)
`s`	Gniazdo (ang. *Socket*)

Tabela 2 Oznaczenia typów pliku

1.7.2 Katalogi i systemy plików

Na typowym dysku z systemem występuje wiele tysięcy plików. Aby ułatwić do nich dostęp pliki zorganizowane są w katalogi. Celem katalogów jest ułatwienie dostępu do plików i nadanie im uporządkowanej struktury.

Implementacja katalogów jest następująca:

- Katalogi są plikami specjalnymi
- Pliki zawierają pozycje odpowiadające plikom lub katalogom.
- Każda z pozycji zawiera co najmniej: nazwę pliku i numer i-węzła. Numer i-węzła jednoznacznie identyfikuje plik w ramach systemu plików.

System plików - zbiór katalogów i plików zorganizowanych w specjalny sposób.

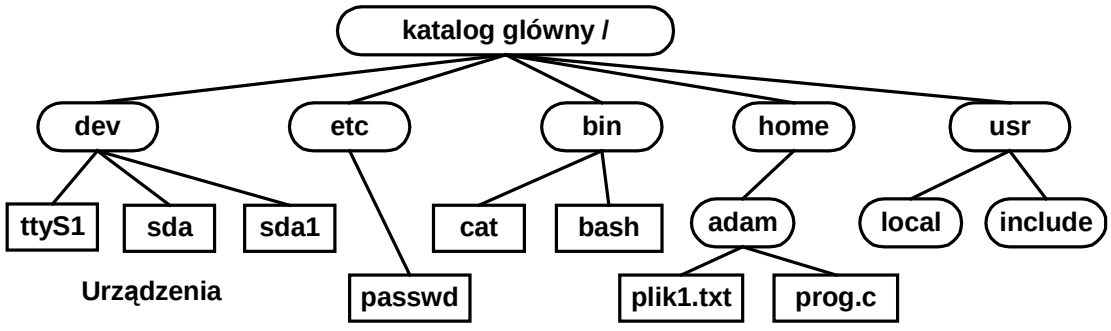
Systemy plików mają budowę hierarchiczną.

Reguły organizacji katalogów tworzących system plików są następujące:

- Katalogi mogą być zagnieżdżane. Każdy podkatalog ma dokładnie jeden katalog macierzysty
- Katalogi są połączone i tworzą strukturę drzewa
- Jeden katalog – katalog główny jest wyróżniony. Tworzy on wierzchołek drzewa katalogów a jego katalogiem macierzystym jest on sam.
- Węzły końcowe są: plikami regularnymi, plikami specjalnymi, katalogami
- W każdym katalogu zawarte są dwie pozycje: . podwójna kropka – jest łączem do katalogu macierzystego, pojedyncza kropka – jest łączem do bieżącego katalogu

Zazwyczaj system plików związany jest z partycją dyskową lub innym urządzeniem pamięciowym.

Katalog ma postać drzewa z wierzchołkiem oznaczonym znakiem



Rys. 0-2 Fragment systemu plików

Położenie określonego pliku w drzewie katalogów określa się za pomocą ścieżki.
Rozróżnia się ścieżki absolutne i relatywne.

- Ścieżka absolutna podaje drogę jaką trzeba przejść od wierzchołka drzewa do danego pliku.
- Ścieżka relatywna zaczyna się od innego znaku niż /. Określa ona położenie pliku względem katalogu bieżącego.

1.7.3 Linux – organizacja systemu plików

Aby posługiwać się systemem trzeba posiadać pewną orientację o organizacji systemów plików. Zawartość i rozmieszczenie systemu plików Linuxa i systemów rodziny Unixa jest przedmiotem standardu FHS (ang. *Filesystem Hierarchy Standard*) **Błąd! Nie można odnaleźć źródła odsyłacza.**.

Standard ten utrzymywany jest przez Free Standard Groups, niedochodową organizację utrzymywaną przez grupę wytwórców sprzętu i oprogramowania. Dzięki standaryzacji niezależnie od mutacji systemu możemy się spodziewać plików w określonym miejscu.