11. Systemy wielodostępne Unix/Linux.

Podobnie jak Unix, Linux może być podzielony na cztery główne części: **jądro**, **powłokę**, **system plików** i **programy użytkowe**. Jądro jest programem, który uruchamia inne programy i zarządza urządzeniami, takimi jak dyski czy drukarki. Powłoka jest interfejsem użytkownika. Odbiera polecenia wydawane przez użytkownika i wysyła je do jądra w celu wykonania. System plików określa porządek, w jakim pliki są zachowane na urządzeniu takim jak dysk. Pliki są umieszczone w katalogach. Każdy katalog może zawierać dowolną liczbę podkatalogów, w których przechowywane są pliki. Jądro, powłoka i system plików stanowią razem podstawową strukturę systemu operacyjnego. Dzięki tym trzem elementom można uruchamiać programy, zarządzać plikami i współdziałać z systemem. Ponadto Linux ma programy użytkowe, które zostały dołączone po to, by zapewnić standardowe cechy systemu. Chociaż istnieje tylko jedna standardowa wersja Linuxa, obecnie jest kilka różnych dystrybucji.

Zarządzanie pamięcią

Podstawowe zagadnienia

Podstawowym zadaniem każdego systemu operacyjnego jest wykonywanie programów. Przed wykonaniem, kod programu i jego dane muszą być umieszczone przynajmniej częściowo w pamięci operacyjnej. Zwykle w każdym systemie dochodzi do sytuacji, gdy zaczyna brakować pamięci dla wykonywanych programów lub nawet dla jednego programu. Z tego względu pamięć operacyjna należy do najważniejszych zasobów każdego systemu komputerowego, a zarządzanie tą pamięcią stanowi jedno z głównych zadań systemu operacyjnego.

Celem zarządzania pamięcią operacyjną jest:

- przydział pamięci fizycznej poszczególnym procesom,
- odwzorowanie logicznej przestrzeni adresowej procesu na fizyczną przestrzeń adresową pamięci,
- ochrona zawartości pamięci,
- współdzielenie obszarów pamięci przez różne procesy.

Przestrzenie adresowe

Adres wytworzony przez procesor w wyniku wykonania rozkazu programu nosi nazwę **adresu logicznego** (ang. logical address). Zbiór wszystkich adresów logicznych generownych przez program tworzy **logiczną przestrzeń adresową** procesu.

Adres fizyczny wskazuje konkretną komórkę pamięci operacyjnej. Zbiór wszystkich adresów fizycznych tworzy **fizyczną przestrzeń adresową**.

Wiązanie adresów polega na odwzorowaniu adresów logicznych generowanych w programie na adresy fizyczne w pamięci operacyjnej. Może następować w dowolnych z poniższych etapów:

- w czasie kompilacji, czyli tworzenia programu,
- w czasie ładowania programu do pamięci,
- w czasie wykonania programu.

Wiązanie podczas kompilacji wymaga dokładnej znajomości początkowego adresu, pod którym program zostanie załadowany do pamięci. Tylko w takiej sytuacji program może posługiwać się bezpośrednio adresami bezwzględnymi w pamięci. W czasie wykonywania programu adresy logiczne i fizyczne są już takie same. Wiązanie podczas kompilacji można zastosować jedynie w prostych systemach przeznaczonych dla jednego użytkownika (np. MS-DOS).

Jeśli początkowy adres w pamięci nie jest znany w czasie tworzenia programu, to wiązanie adresów musi być opóźnione do momentu ładowania programu do pamięci. Po załadowaniu program nie może być już przemieszczany w pamięci podczas wykonywania, ponieważ adresy

logiczne i fizyczne są już takie same. Zmiana adresu początkowego pociąga za sobą konieczność ponownego załadowania całego programu, począwszy od nowego adresu w pamięci.

Największe możliwości swobodnego przemieszczania programu w pamięci daje wiązanie adresów w czasie wykonywania programu. Adresy logiczne muszą być jednak tłumaczone na bieżąco na adresy fizyczne, co wymaga zastosowania specjalnego sprzętu. Zajmuje się tym jednostka zarządzania pamięcią (ang. memory management unit - MMU). Złożoność tej jednostki warunkuje wybór strategii zarządzania pamięcią.

Zwiększenie logicznej przestrzeni adresowej procesu

Ograniczenie logicznej przestrzeni adresowej procesu do rozmiaru pamięci operacyjnej stanowiło przez długi czas poważne ograniczenie w systemach operacyjnych, uniemożliwiając wykonywanie dużych programów. Opracowane zostały dwie metody rozwiązania tego problemu:

- nakładki,
- pamięć wirtualna.

Stosowanie **nakładek** wymaga wyróżnienia w programie kilku funkcjonalnych modułów. Program nie jest w całości wprowadzany do pamięci. W pamięci przechowywane są tylko te moduły, które są stale wykorzystywane. Pozostałe moduły, zwane nakładkami, mogą być wprowadzane zamiennie w miarę potrzeb. Stosowanie nakładek odbywa się w zasadzie bez udziału systemu operacyjnego. Programista musi zadbać o dokonanie odpowiedniego podziału swojego programu na moduły oraz zawrzeć w programie fragmenty kodu odpowiedzialne za ładowanie kolejnych nakładek w odpowiednich momentach. Rozwiązanie to jest zatem bardzo uciążliwe dla programistów. W związku z tym, nie jest obecnie stosowane w systemach operacyjnych ogólnego przeznaczenia działających na typowych komputerach. Polem dla stosowania nakładek pozostają natomiast urządzenia o specyficznym przeznaczeniu dysponujące ograniczonymi zasobami pamięci, takie jak kalkulatory programowalne, komputery kieszonkowe.

Znacznie większe możliwości i większą wygodę oferuje technika **pamięci wirtualnej**. Pamięć wirtualna umożliwia wykonywanie programów, które nie znajdują się w całości w pamięci operacyjnej. Udostępnia procesom dużą ciągłą przestrzeń adresową nieograniczoną rozmiarem fizycznej pamięci operacyjnej. Separuje w ten sposób pamięć logiczną procesu od pamięci fizycznej. System operacyjny zajmuje się odwzorowaniem pamięci wirtualnej używanej przez procesy, częściowo na pamięć operacyjną a częściowo na pamięć pomocniczą.

Ponieważ procesy nie muszą znajdować się w całości w pamięci operacyjnej, każdy z nich zajmuje mniejszy obszar tej pamięci. Dzięki temu w pamięci można umieścić więcej procesów, zwiększając stopień wieloprogramowości. Zmniejsza się liczba operacji wejścia-wyjścia związanych z ładowaniem lub wymianą procesów. W efekcie wzrasta przepustowość systemu.

Linux obsługuje **pamięć wirtualną** - wykorzystuje cześć dysku jako rozszerzenie fizycznej pamięci. Jądro zapisuje zawartość nieużywanych bloków pamięci fizycznej na dysku, umożliwiając tym samym wykorzystanie ich do innych celów. Jeżeli oryginalna zawartość jest potrzebna następuje proces odwrotny. Wszystko to odbywa się niewidocznie dla użytkownika; działające programy również nie dostrzegają różnicy. Oczywiście operacje dyskowe są znacznie wolniejsze (tysiące razy) niż analogiczne działania na fizycznej pamięci, programy zwalniają. Część dysku twardego wykorzystywana jako pamięć wirtualna nosi nazwę **obszaru wymiany**.

Linux potrafi wykorzystywać zwykły plik na systemie plików lub oddzielną partycję jako obszar wymiany. Partycja wymiany jest szybsza, jednak trudniej zmienić jej rozmiar, niż w wypadku pliku. Możesz w celach testowych stworzyć plik wymiany, a gdy już będziesz pewien co do potrzebnego rozmiaru stworzyć osobną partycję.

Powinieneś wiedzieć, że Linux pozwala używać kilku partycji/plików wymiany jednocześnie. Dzięki temu możesz dynamicznie regulować wielkość swapu, np. stworzyć partycję o odpowiednim rozmiarze (takim aby się nie marnowała), a przy specjalnych okazjach wykorzystywać dodatkową partycję/plik, itp.

Krótka lekcja z terminologii komputerowej: komputerowi naukowcy najczęściej rozróżniają miedzy **swapowaniem** (zapisywaniem całego procesu na dysku) a **stronicowaniem** (zapisywaniem części o stałym rozmiarze, zazwyczaj kilku kilobajtów). Stronicowanie jest zazwyczaj szybsze, dlatego Linux go używa, pomimo tego tradycyjna terminiologia Linuxa mówi o swapowaniu.

System plików

System plików tworzy mechanizm bezpośredniego przechowywania i dostępu do informacji w systemie operacyjnym. Odwzorowuje logiczną koncepcję pliku na fizyczne urządzenia pamięci masowej. System Linux może obsługiwać wiele popularnych systemów plików (ext, ext2). Stanowi to jego niewątpliwą zaletę, gdyż umożliwia użytkownikom łatwe posługiwanie się plikami z różnych systemów bez koniecznosci dokonywania kłopotliwych konwersji danych.

Urządzenia

Linux urządzenia postrzega jako pliki. I tak każde urządzenie w Linuksie ma swój ma swój odpowiednik w katalogu **/dev**. Aby odwołać się do jakiegoś urządzenia, system wykorzystuje do tego odpowiedni plik w tym katalogu. Oczywiście nazwy plików są zorganizowane w sposób logiczny i przejrzysty.

Interpretator poleceń, powłoki systemu

Nie można pracować bezpośrednio z podstawową częścią systemu linuksowego, jaką jest jego jądro (określa się je też często nazwą kernel) - niezbędny jest do tego program pośredniczący, czyli właśnie **powłoka systemu operacyjnego** (inaczej **interpreter poleceń powłoki** lub po prostu **shell**). Powłoka systemu Linux pełni taką samą funkcję, jak plik command.com w systemie DOS, tyle tylko, że użytkownik Linuksa może wybrać jedną spośród kilku dostępnych powłok. Domyślna powłoka systemów linuksowych to **/bin/bash**.

Powłoka systemu operacyjnego to program, który udostępnia interfejs pomiędzy użytkownikiem a jądrem systemu; ma on postać wiersza poleceń. Jądro systemu zawiera wszelkie podprogramy potrzebne do przeprowadzania operacji wejścia i wyjścia, zarządzania plikami itp. Powłoka pozwala korzystać z tych podprogramów za pomocą wiersza poleceń. Poza tym, powłoki obsługują również język programowania. Programy napisane w języku powłoki nazywane są zwykle skryptami lub skryptami powłoki.

Interpretator poleceń będący interfejsem między użytkownikiem a SO jest albo zawarty w jądrze SO, lub jak w DOS (command.com), Linuksie (bash), jest specjalnym programem wykonywanym przy rozpoczęciu zadania lub zalogowaniu się. Program interpretujący instrukcje sterujące w Unix-ie to powłoka (shell).

Polecenia rozpoznawane przez interpreter dotyczą:

- tworzenia procesów i zarządzania nimi,
- obsługa WE/WY,
- administrowanie pamięcią pomocniczą i operacyjną,
- dostępu do plików,
- ochrony,
- pracy sieciowej.

Praca w tle

PID - każdy proces ma przyporządkowany unikalny numer. Numer ten zwany PID-em zostaje nadany mu podczas uruchamiania. Dzięki temu proces jest identyfikowany w systemie.

Czasami gdy poszukujemy jakiś plików w systemie bądź kompilujemy coś, trwa to dosyć długo. Za pomocą polecenia **bg...&** można wysłać wykonanie zadania do pracy w tle, co zwolni nam jednocześnie terminal. Na ekranie wyświetla nam się wtedy pid tego zadania. Za pomocą

polecenia **jobs** można sprawdzić identyfikator zadań, a później za pomocą **%** można zastopowane zadania (**exit**) wysłać w tło lub na pierwszy plan np. **bg %3**. Polecenie **fg** przywróci zadanie na nasz terminal (**fg % indentyfikator)** np. **fg %3**.

Licencja GNU

W odróżnieniu od oficjalnego systemu operacyjnego Unix, Linux jest rozpowszechniany za darmo na zasadach licencji GNU, określanej przez Free Software Foundation. Wprawdzie Linux ma zastrzeżone prawa autorskie i nie jest oprogramowaniem ogólnie dostępnym (Public Domain), jednak licencja GNU sprawia, że może z niego korzystać każdy. Licencja ma zapewnić, że Linux pozostanie darmowy i zarazem zestandaryzowany. Istnieje tylko jeden oficjalny Linux.

Konsole wirtualne, terminale, zdalne konsole

Obecnie wbudowana obsługa terminali w GNU/Linuksie umożliwia zarówno zdalną pracę (np. zdalną administrację serwerem), jak i wykorzystanie wielozadaniowości systemu także w środowisku tekstowym (uruchomienie wielu terminali odpowiada funkcjonalnie otwarciu wielu okien, gdzie w czasie gdy jeden program jest zajęty i nie odpowiada, możemy pracować w innym).

Linux domyślnie posiada siedem wirtualnych konsol, do których mamy dostęp za pomocą kombinacji klawiszy : ALT+F1-F7. Podczas uruchamiania systemu X Window ręcznie lub automatycznie, zablokowana zostaje konsola, z której zostały uruchomione X-y, a nasz menadżer wyświetlania dostępny jest na konsoli 7. Przejście z poziomu X-ów na pozostałe konsole odbywa się za pomocą następującej kombinacji klawiszy : CTR+ALT+F1-F6.

Na każdej z wirtualnych konsol możemy zalogować się jako inny użytkownik. Takie rozwiązanie umożliwia nam uruchomienie na każdej z konsol jakiegoś zadania. Należy jednak pamiętać, że Linux to prawdziwy system wielozadaniowy, który umożliwia przeniesienie dowolnego procesu w tło, zwalniając terminal i ponowne przywrócenie go w dowolnym momencie.

Użytkownicy, grupy, logowanie, uwierzytelnianie

W Linuksie bardzo ważne jest zagadnienie ***uprawnień*** - każdy plik, każdy uruchomiony proces, musi mieć pewne określone uprawnienia. Uprawnienia te określają, co dany proces może wykonywać, i co możemy z każdym plikiem zrobić (obejrzeć, zapisać).

W systemie można utworzyć wiele ***kont użytkowników*** - jeżeli jesteśmy zalogowani na konto danego użytkownika, to mamy pewne określone uprawnienia. Zawsze jest tworzony *superużytkownik* **root** - jest to konto, z którego mamy dostęp do wszystkich plików, niezależnie od ich uprawnień. Konto to jest przeznaczone tylko i wyłącznie do wykonywania zadań związanych z administrowaniem systemem (instalacja/usuwanie oprogramowania, zmienianie ustawień, rozwiązywanie problemów, naprawianie systemu, tworzenie użytkowników i przyznawanie im uprawnień).

Każdy plik i katalog w linuksowym systemie pliku (ext3, lub - dawniej - ext2) ma dodatkowe właściwości - ***prawa dostępu***. Określają one, którzy użytkownicy mają do tego pliku/katalogu dostęp.

Jądro, moduły ładowalne, obsługa urządzeń

Jądro w Linuksie

Jądro systemu jest programem napisanym w języku C. Dostępne jest ono w postaci kodu źródłowego lub skompilowanych pakietów binarnych. Pakiety możemy wykorzystać od razu, lecz ze źródeł sami musimy zbudować własne jądro systemu. Do zbudowania jądra niezbędny jest **kompilator**, który tłumaczy program napisany w języku programowania na język zrozumiały dla komputera. Przekompilowanie jądra dostarczonego z dystrybucją pozwala na zwiększenie szybkości działania całego systemu. Natomiast jądra nowszej wersji posiadają poprawione błędy znalezione w wersji wcześniejszej i często zwiększoną ilość sterowników i funkcji.

Jądro systemu operacyjnego Linux stanowi jego podstawę. Znajdują się tam główne sterowniki i programy odpowiadające za działanie systemu. Każdy system komputerowy posiada zbiór programów, który nazywany jest systemem operacyjnym. Najważniejszym z tych programów jest jądro (kernel). Jądro jest ładowane do pamięci RAM w czasie uruchamiania systemu, zawiera ono wiele niezbędnych dla systemu procedur. Kształt i możliwości komputera opierają się na jego jądrze. Często słowa "system operacyjny" używa się w odniesieniu do jądra systemu. Jądro współdziała ze sprzętem poprzez programy niskiego poziomu. Dostarcza też środowisko dla aplikacji działających w systemie. Kiedy program chce wykorzystać zasoby sprzętowe, musi wystosować odpowiednie zapytanie. Jądro rozważa to zapytanie i wybiera czy, jak i kiedy użyczyć programowi potrzebnych mu zasobów.

Jądro w systemach typu Unix odgrywa rolę pośrednika między programami, a sprzętem. Najpierw zajmuje się przydziałem pamięci dla wszystkich uruchomionych programów (procesów), i dba o to, aby wszystkie one dostały równą ilość czasu procesora.

Rodzaje wersji jądra

Istnieją dwa rodzaje dostępnych wersji jądra - **stabilna** (stable) i **rozwojowa** (development). Stabilna przeznaczone jest dla ludzi ceniących sobie niezawodność i stabilność systemu jak również jego bezproblemową obsługę. Natomiast wersje rozwojowe (developerskie) przeznaczone są dla ludzi zajmujących się rozwojem jądra, zawierają one często wiele nowych sterowników dla najnowszych urządzeń i wiele funkcji eksperymentalnych, które mogą zniknąć w następnym jądrze. Jądra te mogą także być niestabilne i powodować liczne problemy.

Pierwsza liczba oznaczenia jądra przedstawia numer wersji, następna czy jądro jest stabilne (liczba parzysta) czy rozwojowe (liczba nieparzysta), ostatnia liczba określa natomiast numer wydania.

Moduły

Moduły są to części jądra, które nie są zawarte bezpośrednio w nim. Kompiluje się je osobno i można je umieścić, a następnie usunąć z uruchomionego jądra prawie zawsze. Z powodu tej elastyczności jest to teraz preferowana metoda pisania niektórych fragmentów jądra. Wiele popularnych sterowników urządzeń to ładowalne moduły. Są to sterowniki różnych urządzeń, które nie są wkompilowane bezpośrednio w jądro. Jednak gdy zajdzie potrzeba użycia takiego sterownika zostaje on wtedy załadowany dynamicznie przez specjalny program ładujący bez przerywania pracy naszego systemu.

Wszystkie moduły dostępne w naszym systemie znajdują się w katalogu: /lib/modules/numer_wersji_naszego_jądra.

Korzyść z użycia modułów jest taka, że przez ich zastosowanie otrzymujemy mniejsze i szybsze jądro, dzięki czemu nie trwonimy bez potrzeby zasobów naszego komputera, w razie gdy potrzebujemy raz na jakiś czas skorzystać z sterownika a mamy go jako moduł, możemy załadować go na czas pracy, a gdy już nie będzie nam potrzebny spokojnie możemy skasować go z pamięci. Jednak nie wszystkie sterowniki możemy użyć jako moduły, Linux potrzebuje część z nich jeszcze zanim zostaną one załadowane przez system, tak jest np. z obsługą dysku twardego czy systemu plików, aby system mógł się poprawnie uruchomić i obsłużyć posiadany przez nas sprzęt.

Obsługa urządzeń w Linuksie

Obsługa urządzeń zewnętrznych w Linuksie dzieli je na dwie kategorie:

1. urządzenia znakowe,
2. urządzenia blokowe.

W grupie **blokowych** znajdą się te urządzenia, które pozwalają na swobodny dostęp do bloków danych (Random Access). W grupie tej mieszczą się dyski twarde, stacje CD-ROM czy FDD. Pliki obsługujące te urządzenia posiadają wszelkie własności plików regularnych.

W grupie **znakowych** znajdzie się natomiast szereg urządzeń, które nie pozwalają na swobodny odczyt/zapis. Przykładem może być choćby urządzenie obsługujące kartę dźwiękową (/dev/audio). Odczytując z niego dane dostaniemy to co jest aktualnie podawane na wejście karty dźwiękowej (np. dźwięk zebrany przez mikrofon).

Warto zauważyć, że takie podejście do obsługi urządzeń jest bardzo wygodne. Polecenia, które nie mieszczą się w standardzie obsługi urządzenia (np. odczytanie liczby ścieżek czy głowic twardego dysku, zmiana częstotliwości próbkowania pobieranych czy odczytywanych z karty muzycznej danych, otwieranie kieszeni CD-ROMu) realizowane są przez przesłanie do sterownika urządzeń polecenia nazywanego ioctl.

Tworzone urządzenia nie muszą być związane z fizycznym sprzętem. Program obsługi może zwracać dowolne dane (takimi urządzeniami są np. /dev/null, /dev/zero, w starszych wersjach jądra /dev/sndstat).