System dwójkowy binarny

2 – 010, 3-011, 4-100, 5-101, 6-110, 7-111

Podstawowe warunki i przyczyny powstania systemu operacyjnego Linux:

- wąskie i specjalistyczne grono użytkowników systemów uniksowych
- szybki rozwój internetu na początku lat 90
- powstanie procesora Intel 80386 umożliwiającego pracę wielozadaniową
- napisanie przez Linusa Torvaldsa programów dla systemu operacyjnego Minix oraz jądra nowego systemu operacyjnego (opartego na Miniksie)
- ogłoszenie przez Torvaldsa na grupie dyskusyjnej comp.os.minix (sierpień 1991) prac nad bezpłatnym systemem dla komputerów PC
- dostosowanie systemu Linux do norm POSIX pozwalające na przenoszenie oprogramowania

pomiędzy Linuksem, a komercyjnymi systemami uniksowymi

- opracowanie Linuxa na podstawie licencji GNU GPL (GNU General Public Licence) fundacji FDF (Free Software Fundation)
- pojawienie się różnych dystrybucji Linuxa (np. Slackware, RedHat, Debian, Ubuntu, SuSE)

Dystrybucja systemu Linux to

System operacyjny to program lub zbiór programów służących do zarządzania pracą komputera. Linux jest systemem wielozadaniowym (tryb pracy systemu operacyjnego, w którym użytkownik może uruchomić w tym samym czasie wiece jniż jedno zadanie) i wielodostępowym (tryb pracy systemu operacyjnego, w którym w tym samym czasie więcej niż jeden użytkownik może pracować z systemem i korzytać z jego zasobów).

Równoczesna praca jest możliwa za pomocą terminali.

Praca zdalne w trybie tekstowym, może być realizowana za pomocą protokołu Telnet lub SSH – oba zapewniają wielodostęp, ale SSH pracuje na szyfrowanym tekście, a telnet na jawnym.

Do pracy zdalnej z zastosowaniem SSH mogą być wykorzystywane progr. CRT, SSh32, Putty

---

System plików jest to sposób zapisu **logicznej struktury danych** na fizycznym nosniku.

System plikow = zbior plików – sposób ich uporzadkowania

Pliki, jako podstawowe jednostki logiczne systemu plików:

- służą do przechow. programów i danych
- są przechowywane na nośnikach danych
- mają strukturę ustaloną przez ich twórcę
- posiadają swoją nazwę i atrybuty

- są zarządzane przez system operacyjny w sposób zgodny z wymaganiami i możliwościami systemu plików

  Funkcje systemu operacyjny w zakresie zarządzania systemem plików:

- przydział miejsca na nośniku dla systemu plików oraz ewidencja zajętych i wolnych obszarów nośnika
- określenie sposobu uporządkowania plików i organizowanie do nich dostępu zgodnie z wymogami i możliwościami systemu plików
- wykonanie operacji na plikach (kopiowanie, przesuwanie, kasowanie, zmiana nazwy)

  W systemie plików Linuxa katalog:

  /bin zawiera niezbędne do funkcjonowanie systemu programy

  /boot zawiera jądro systemu i pliki startowe

  /dev zawiera urządzenia widziane jako pliki

/etc zawiera pliki konfiguracyjne systemu

/home zawiera katalogi domowe użytkowników

/lib zawiera m.in. biblioteki systemowe i ładowalne moduły jądra

/lost+found zawiera odzyskane pliki

/mnt zawiera zamontowane systemy plików innych urządzeń np. dyskietki, cd-romu

/proc zawiera pliki statusu jądra, urządzeń i procesów

/root katalog domowy administratora

/sbin zawiera programy wykorzystywane przez administratora do zarz. i konfigurowania systemu

/tmp zawiera pliki tymczasowe tworzone przez różne programy podczas ich pracy

/usr zawiera programy, biblioteki i dokumnety dostępne dla użytkowników systemu

/var zawiera przychodząca i wychodzącaą pocztę oraz logi systemowe

Każdy plik posiada swoją nazwę o długości do 255 znaków. Nazwa może składać się z liter, cyfr, znaku podkreślenia, spacji, kropek. Nazwa zaczynająca się od kropki oznacza plik ukryty. W nazwach rozróżniane są małe i duże litery, zazwyczaj stosuje się tylko małe w nazwach plików i katalogów.

znak slash / symbolizuje katalog główny

kropka . symbolizuje katalog bieżący

dwie kropki .. symbolizują katalog nadrzędny

znak tyldy ~ symbolizuje katalog domowy użytkownika

Położenie pliku w systemie określa ścieżka dostępu.

- bezwzględna, czyli taka, która określa położenie pliku względem katalogu głównego

home/krysia/grafika/rys.jpg

- względna, czyli taka, która określa położenie pliku względem bieżącego katalogu
  ../grafika/rysunek.jpg

Wyszukiwanie informacji o poleceniach:

- apropos (składnia apropos szukany_ciąg_znaków)
  program przeszukuje opisy poleceń np.
  apropos „copy files” wyszukaj poleceni, wk tóirych opisie znajduje się ciąg znaków „copy files”
  apropos password | less – wyszukaj polecenia, w których opisie znajduje się w słowo password
- man, info lub opcja –help

Podstawowe polecenia systemu Linux:

- **pwd** wyświetlanie nazwy bieżącego katalogu
- **ls** wyświetlanie zawartości katalogu (-tl = pokaż czas jako jedno słowo, -t use long listing format)
- **cd** zmiana bieżącego katalogu

- polecenie **mkdir** utworzenie katalogu
- polecenie **tree** wyświetlenie struktury katalogów i plików
- polecenie **rmdir** usunięcie pustego katalogu
- polecenie **find** wyszukiwanie plików i katalogów zgodnie z zadanym kryterium
  find / -name index.txt znajdz w kat. gł. i podkat. wszystkie pliki o nazwie index.txt
- polecenie **cat** wyświetlenie zawartości pliku
- polecenie **more** wyświetla ilość tekstu mieszczącą się na jednej stronie, a następnie umożliwia przewijanie go w przód
- polecenie **less** umożliwa przewijanie tekstu w przód i w tył
- polecenie **rm** usuwanie plików /rm –rf usuwa katalogi z zawartością (-r) ignoruje nieistniejące pliki (-f)
- polecenie **cp** kopiowanie plików

- polecenie **mv** przenoszenie plików (zmiana nazwy)
- polecenie **echo** wyświetlnenie na ekran tekst lub wartości zmiennych
- **$?** odwołanie się do kodu wyjścia ostatnio wykonanego polecenia
- **[tab]** uzupełnia nazwy plików
- **strzałki góra dół** przeskoczenie do wydanych wcześniej poleceń
- polecenie **history** przeglądanie historii poleceń
- polecenie **du** wyświetlanie informacji o zajętości katalogów
- polecenie **df** wyświetlanie informacji o zajętości dysków
- polecenie **quota** wyświetlanie informacji o ograniczeniach użytkownika co do ilości miejsca i liczby utworzonych plików

---

Edytor tekstów VIM

Nazwa edytora vim oznacza rozbudowany edytor vi (vi improved). Edytore ten został napisany przez Brama Moolenaara, holenderskiego programistę (pierwsza wersja w 1991 roku) i jest klonem edytora tekstu vi. Obecnie w zdecydowanej większości dystrybucji Linuksa wydanie polecenia vi powoduje uruchomienie edytora vim.

Zalety edytora vim:

- popularność – standardowo dostępny edytor w systemach uniksowych
- możliwośc pracy z bardzo dużymi plikami tekstowymi
- podświetlanie składni wielu róznych języków programowania
- zaawansowane wyszukiwanie tekstu
- łatwe pisanie makropoleceń

Tryby pracy edytora vim:

- normalny tryb wydawania poleceń

- tryb wprowadzania tekstu
- tryb zastępowania tekstu
- tryb wydawania poleceń z linii poleceń (tzw. tryb poleceń)

  Normalny tryb wydawania poleceń (tryb normalny) jest to tryb, który służy do:

- poruszania się po dokumencie tzn. przeglądanie tekstu oraz proste wyszukiwanie znaków
- zaawansowane usuwanie fragmentów tekstu
- kopiowanie fragmentów tekstu do bufora oraz wstawianie ich do dokumentu

  Tryb wprowadzania tekstu pozwala wprowadzać tekst do dokumentu oraz go usuwać. Przejście do trybu wprowadzania następuje po użyciu w trybie normalnym jednego z takich poleceń jak np. i, I, a, A, o, O. Powrót z trybu wprowadzania tekstu do trybu normalnego – klawisz ESC

Tryb zastępowania tekstu pozwala na nadpisywanie wprowadzonego tekstu nowym. Przejście do trybu zastępowania następuje po użyciu w normalnym trybie polecenia R. Powrót z trybu zastępowania do trybu normalnego – ESC.

Tryb wydawania poleceń z linii poleceń (tryb poleceń) pozwala na:

- zaawansowane wyszukiwanie i zamianę znalezionego tekstu
- operacje na plikach i wydawanie poleceń systemowych
- zmianę sposobu wyświetlania tekstu (np. numerowanie wierszy tekstu) oraz konfiguracje edytora (określenie zasad traktowania dużych i małych liter podczas wyszukiwania tekstu).

Przejście do trybu wydawania poleceń następuje po naciśnięciu w normalnym trybie dwukropka (:) Powrót z trybu wydawania poleceń do trybu

normalnego następuje automatycznie po wykonaniu polecenia.

VIM:

j kursos o jeden wiersz w dół

k kursos o jeden wiersz w górę

h kursor o jeden znak w lewo

l kursor o jeden znak w prawo

L kursor na dół ekranu

H kursor na górę ekranu

M kursor na środek ekranu

CTRL+f przesunięcie kursora o jeden ekran w dół

CTRL+ b kursor o jeden ekran w górę

CTRL+d o pół ekranu w dół

CTRL+u o poł ekranu w górę

G przejście na poczatek ostatniego wiersza

nG przesunięcie kursora do n-tego wiersza

$ przesunięcie kursora na koniec bieżącego wiersza

^ przesunięcie kursora na początek bieżącego wiersza

fx przesunięcie kursora do najbliższego znaku x znajdującego się po prawej stronie kursora w bież.wie

Fx przesunięcie kursora do najbliższego znaku x znajdującego się po lewej stronie kursora w bież.wie

; powtórzenie ostatniego polecenia f lub F

w przesuniecie kursora o jeden wyraz do przodu

nw przesuniecie kursora o n wyrazów do przodu

b przesuniecie kursora o jeden wyraz do tyłu

nb przesuniecie kursora o n wyrazów do tyłu

e przesuniecie kursora na koniec bieżącego wyrazu

(przesuniecie kursora o jedno zdanie do tyłu

) przesuniecie kursora o jedno zdanie do przodu

KOŃCZENIE PRACY:

:q zakończenie pracy edytora

:q! zakończenie pracy edytora bez zapisywania zmian

:wq zakończenie pracy edytora z zapisaniem zmian

:r nazwa_pliku wstawienie zawartości pliku w bieżącym wierszu

:! wykonanie poleceń powłoki

:r! wstawienie wyniku działania polecenia w bieżącym wierszu

KOPIOWANIE:

yw skopiowanie tekstu od bieżącej pozycji kursora do końca wiersza

yb skopiowanie tekstu od początku wyrazu do bieżącej pozycji kursora

y$ skopiowanie tekstu do bieżącej pozycji kursora od końca wiersza

Y skopiowanie do bufora bieżącego wiersza

yy działa tak jak Y

p wklejenie zawartości bufora za znakiem wskazywanym przez kursor

P wklejenie zawartości bufora przed znakiem wskazywanym przez kursor

OBSŁUGA REJESTRÓW:

reg: wyświetla listę i zawartość wszystkich używanych rejestrów edytora

„{rejestr} wykonuje operacje na określonym rejestrze:

„ aY skopiuje wiersz tekstu do rejestru
„ap wklei zawartość rejestru a

„bdd usunie zawartość bieżącego wiersz (polecenie dd i umiesci w rejestrze b)

q{rejestr} rozpoczyna nagrywanie makra do wskazanego rejestru

qa rozpocznie nagrywanie makra i umieście je w rejestrze a

q kończy nagrywanie makra

@{rejestr} -> @a wykona makro z rejestru a

PISANIE SKRYPTÓW:

. powtarza ostatnio wydane polecenie

Ctrl + N dopełnia słowo

]p wkleja tekst nad wierszem, gdzie znajduje się kursor, ale automatycznie wyrównuje poziom wcięcia wklejonego kodu dopasowując go do tego, w którym jest kursor

]p wkleja tekst pod wierszem, gdzie znajduje się kursor, ale automatycznie wyrównuje poziom

wcięcia wklejonego kodu dopasowując go do tego, w którym jest kursor

% po ustawieniu kursora na nawiasie naciśnięcie % powoduje przeniesienie kursora do odpowiedniego nawiasu w parze nawiasów

n>> robi wcięcie dla n kolejnych wierszy zaczynając od tego, gdzie jest kursor

gd idzie do deifnicji funkcji lub zmiennej pod kursorem. kolejne wydanie polecenia n powoduje przejście do kolejnych wystapień tej funkcji lub zmiennej

m[litera} ustawia w tekście zakładkę dostepną pod wskazaną literą np. mb

‘{} przejście do ustawionej zakładki

‘. przechodzi do ostatnio edytowanego wiersza

K – idzie do strony podręcznika polecenia systemowego znajdującego się pod kursorem

:help polecenia wyświetla pomoc na temat wskazanego polecenia edytora vim

:syntax on włącza podświetlanie składni

WIELE OKIEN:

[CTRL+W] j/k przejście do okna poniżej /powyżej

[CTRL+W] t/b przejście do górnego / dolnego okna

:help wyświetlenie w nowym oknie pomocy

:new utworzenie nowego okna edytora

:split/[CTRL+W] podział aktualnego okna na 2 części

WPROWADZANIE:

i rozpoczęcie wprowadzania tekstu od bieżącej pozycji kursora

a rozpoczęcie wprowadzania tekstu od znaku znajdującego się za kursorem

A rozpoczęcie wprowadzania tekstuna końcu linii gdzie znajduje się kursor

O wstawienie nowego wiersza powyżej kursora i rozpoczęcie wprowadzania tekstu w nowym wierszu

o wstawienie nowego wiersza poniżej kursora i rozpoczęcie wprowadzania tekstu w nowym wierszu

WYSZUKIWANIE:

/abc wyszukiwanie ciągu abc w dół ekranu

?abc wyszukiwanie ciągu abc w górę ekranu

n powtórzenie ostatniego polecenia wyszukania ciągu znaków w dół ekranu

N powtórzenie ostatniego polecenia wyszukania ciągu znaków w górę ekranu

:set ic/ :set noic włączenie wyłączenie trybu ignorowania wielkości znaków podczas wyszukiwania tekstu

USUWANIE:

x usuniecie znaku znajdującego się pod kursorem

X usuniecie znaku znajdującego się przed kursorem

de usuniecie znaków od miejsca wskazywanego przez kursor do końca bieżącego wyrazu

dw usuniecie znaków od miejsca wskazywanego przez kursor do początku następnego wyrazu

d$ usuniecie znaków od bieżącej pozycji kursora do końca wiersza

dG usuniecie znaków od bieżącej pozycji kursora do końca pliku

dnh usuniecie n znaków poprzedzających

dnl usuniecie n znaków kolejnych

D usuniecie znaków do końca bieżącego wiersza (wiersz zostaje)

J połączenie wiersza poniżej kursora z wierszem bieżącym

dd usunięcie bieżącego wiersza i znaków w nim

dnd usunięcie bieżącego wiersza i n-1 kolejnych wierszy

u anulowanie wprowadzonych zmian (m.in. usuniętego tekstu)

1GdG usuń wszystko

ZAMIANA:

:s/stary/nowy znalezienie w bieżącym wierszu pierwszego wystąpienia tekstu stary i zastąpienie go tekstem nowy

:n,m s/stary/nowy/g przeszukanie tekstu od wiersza n do wiersza m i znalezienie w tym

obszarze wszystkich wystąpień tekstu stary i zastąpienie go tekstem nowy

:1,$ s/stary/nowy/g globalne zastępowanie w całym pliku

:1,$ s/stary/nowy/gc globalne zastępowanie – każda operacja zamiany wymaga akceptacji przez użytkownika

ZASTĘPOWANIE:

rx zastąpienie znaku znajdującego się pod kursorem znakiem „x”, po dokonaniu zmamiany edytor pozostaje w trybie wydawania poleceń

cw usunięcie znaków do końca biezącego słowa i przejście do trybu wstawiania tekstuu

C usunięcie znaków do końca wiersza i przejście do trybu wprowadzania

R włączenie trybu zastępowania

Bufor edytora

Edytor tekstu vim umożliwia umieszczenie fragmentu tekstu do bufora, a następnie wklejenie go w dowolone miejsce dokumentu. W buforze umieszczany jest fragment tekstu, który został skopiowany lub usunięty. Tekst bufora może być wielokrotnie wstawiany do dokumentu.

Koniec pracy z edytorem

Do zapisani zmian w dokumencie i zakończenia pracy z edytorem vim służy polecenie rwq. Jeżeli użytkownik nie chce zapisywać wprowadzonych zmian musi wydać polecenie :q!

Konfiguracja edytora vim

Ustawienia edytora zapisywane sąw plik .vimrc, który musi znajdować siew katalogu domowym użytkownika. Najczęściej zapisywane ustawienia konfiguracji :

- **syntax on** włączenie podświetlania składni edytowanego kodu programu
- **set number** numerowanie wyświetlanych wierszy dokumentu
- **set nonumber** wyłączneie numerownia wyświetlanych wierszy
- **set ts=4** ustalenie pozycji tabulacji na 4 znaki
- **set textwidth=80** ustawienie długości wyświetlanego wiersza na 80 znaków
- **set autoindent** włączenie autowcięcia
- **set noautoindent** wyłączenie autowcięcia
- **set incsearch** automatyczne wyszukiwanie ciągu znaków podczas korzystania z polecenia „/”
- **set ruler** wyświetlenie współrzędnych kursora
- **set showmode** wyświetlenie informacji o trybie edytora
- **set showcmd** wyświetlenie wydawanego polecenia
- **set backup** tworzenie kopii zapsowych plików

- **set backupid=~/backup/** określenie katalogu, w którym tworzone będą kopie zapasowe

---

Skrypty powłoki

Cechy skryptów:

- skrypt systemowy (skrypt powłoki) to program napisany z wykorzystaniem poleceń systemu operacyjnego, konstrukcji programistycznych dostepnych w danej powłoce oraz programów zewnętrznych. Ma on postać pliku tekstowego, a jego wykonanie polega na realizacji poleceń zapisanych w kolejnych wierszach pliku.
- W miejscach zakończenia skryptu powinno być polecenie exit wraz z numerem tzw. kodu wyjścia = exit code **jest to odpowednik error level** z systemu DOS/Widonws. Wartość 0 symbolizuje brak błędów, czyli sygnał pomyślnego zakończenia. Inne wartości oznaczją wystąpienie błędów (max. wart. 255)

- Wiersz, który zaczyna się odd znaku # traktowany jest jako komentarz i nie jest wykonywany.
- Pierwsza linia skryptu powinna mówić o tym, dla jakiego rodzaju powłoki został napisany skrypt (#!/bin/bash)
- jeśli napisany skrypt ma być programem powłoki to musi mieć nadane prawo do wykonania
- rozszerzenie pliku zawierającego skrypt to SH, ale może go nie być

  Skrypt można uruchomić poprzedzając jego nazwę kropką i spacją lub przekierowując jego zawartość do polecenia bash.

  Skrypt powłoki ma dla użytkownika status PROGRAMU POWŁOKI wtedy, kiedy użytkownik ma nadane prawo do jego wykonywania. (chmod u+x plik.sh)

  Uruchamianie programu jest możliwe poprzez podanie jego nazwy tj plik.sh, ale taki sposób uruchomienia jest możliwy TYLKO wtedy, gdy

zmienna PATH zawiera ścieżkę dostępu do katalogu bieżącego. Jeśli PATH nie przeszukuje bieżącego katalogu, to taki skrypt należy uruchamiać przez względną ścieżkę dostępu („./”)v czyli ./kopia .sh

---

Parametry skryptu

Parametr skryptu to informacja przekazywana do skryptu w momencie jego uruchomienia. Największą zaletą tworzenia skryptów z parametrami jest możliwośc podstawienia w ich miejsce różnych wartości. Przykładowo, skrypt o nazwie dopisz.sh, który do pliku o nazwie podanej jako pierwszy parametr dopisuje zawartość pliku o nazwie podanej jako drugi parametr, może być wykonany z takimi parametrami:

dopisz.sh list1.txt list2.txt

co oznacza, że dopisze do zawartości pliku list.txt zawartość pliku list2.txt. Uogólniając można

powiedzieć, że wywołanie skryptu ma następującą postać:

dopisz.sh parametr1 parametr2

gdzie parametr1 i parametr2 są nazwami plików.

Parametry podawane przy urucomieniu skryptu nazywane są PARAMETRAMI AKTUALNYMI.

Aby skrypt mógł wykonać postawione zadanie musi mieć możliwość odwołania siędo wartości parametrów podanych przy jego uruchomieniu. Możliwość odwołania się daje wykorzystanie symboli, w miejsce których, w czasie wykonywania skryptu, wstawiane są wartości parametrów aktualnych.

- **$1** symbolizuje pierwszy parametr aktualny skryptu
- **{$10}** symbolizuje dziesiąty parametr skryptu, od 10 trzeba używać nawiasów klamrowych

- **$10** wyświetli wartość parametru 1 z dodatkowym zerem
- **$0** symbolizuje nazwę skryptu, ale tylko takiego, który jest uruchamiany jako program powłoki wraz ze ścieżką dostępu podaną w poleceniu lub uzyskaną ze zmiennej PATH
- **$#** symbolizuje liczbęparametrów aktualnych
- **$@** symbolizuje wszystkie parametry skrytpu, jako osobne ciągi znaków
- **„$*”** symbolizuje łącznie wszystkie parametry aktualne, z którymi został uruchomiony skrypt. Sa one traktorwane jako jeden napis, ale tylko wtedy, gdy zmienna ujęta jest w cudzysłowy. W przeciwnym wypakdu działanie jest takie samo jak $@
- **$?** wartośc zwrócona przez ostatnie polecenie (exit code)
- **$$** zwraca PID bieżącego procesu

- **$_** zwraca ostatni arument poprzedniego polecenia
- **$!** zwraca PID ostatniego polecenia uruchomionego w tle. Zmienna będzie pusta, jeśli powłoka b ash nie wykonanała żadnego polecenia w tle

  Parametry w tekście skryptu, symbolizujące parametry aktualne, nazywane są PARAMETRAMI FORMALNYMI.

  Rejestry edytora

  Edytor vim posiada system rejestrów (schowków) przeznaczonych do przechowywania fragmentów tekstu, w tym zestawów poleceń, które mogą być traktowane jako makropolecenia wykonywane. Rodzajów rejestrów jest kilka (np. rejestry anonimowe /nienazwane/, nazwane, ponumerowane, małych usunięć, wyszukiwania, wyrażeń, tylko do odczytu)

Rejestrów nazwanych jest 26 i są one symbolizowane przez małe litery alfabetu.

Podobnie jak rejestry nazwane działają rejestry numerowane, ale użytkownik nie może wskazać, z którego rejestru chce korzystać. Rejestry te oznaczone cyframi od 0 do 9 o przechowują wyniki dziesięciu ostatnich operacji kopiowania i usuwania, ale tylko wykonanych na przynajmniej jednej linii tekstu oraz takich, dla których użytkownik nie wskazał rejestru nazwanego. Wynik ostatniej operacji jest zapisywany w rejestrze 0, a wcześniej zapisane wynik są przesuwane do kolejnych rejestrów.

Operacje wejścia-wyjścia

Strumień danych oraz urządzenia wejścia i wyjścia:

- dostarczanie danych do procesów oraz generowanie danych przez procesy polega na przesyłaniu tzw. STRUMIENI DANYCH
- dane do procesu trafiają jako STRUMIEŃ DANYCH WEJŚCIOWYCH, wygenerowany przez URZĄDZENIE WEJŚCIA (klawiatura, plik), a z procesu wychodzą jako STRUMIEŃ DANYCH WYJŚCIOWYCH skierowany do URZĄDZENIA WYJŚCIA (monitor, plik, drukarka)
- w przypadku urządzeń wejścia wyjścia wyróżnia się STANDARDOWE URZĄDZENIA wyjścia i wejścia
  - stdin (standardowe wejście) to urządzenie, z którego proces otrzymuje domyślnie strumień danych, zazwyczaj klawiatura
  - stout (standardowe wyjście) to urządzenie, do którego proces domyślnie kieruje strumień danych, zazwyczaj monitor

  Przekierowania wejścia i wyjścia:

- przekierowanie wejścia oznacza zmianę urządzenia wejściowego Z DOMYŚLNEGO NA INNE, wskazane przez użytkownika. służy do tego operator <
- przekierowanie wyjścia oznacza zmianę urządzenia wyjściowego Z DOMYŚLNEGO NA INNE, wskazane przez użytkownika. służy do tego operator > (zastąpienie) lub >> (dopisanie)

  Standardowe wyjście błędów to urządzenie (domyślnie monitor), gdzie trafia strumień danych, który zostaje wygenerowany, kiedy polecenie zakończyło sięniepowodzeniem. Do przekierowania standardowego wyjścia błędów służy operator 2>

  Identyfikatory standardowego wejścia i wyjść

  W operacjach przekierowania strumienia:

- standardowe wyjście jest identyfikowane przez „&1”

- standardowe wyjście błędu jest identyfikowane przez „&2”

  Deskryptory plików

  Z pojęciem strumienia danych jest także związane pojęcie deskryptor pliku. Jest to identyfikator używany przez system operacyjny do obsługi operacji wejścia i wyjścia.
  W standardzie POSIX deskryptor pliku jest liczbą całkowitą (typu int z języka C/C++), a tablica deskryptorów plików jest odrębna dla każdego procesu (czyli kazdego uruchomionego programu). Każdy proces po uruchomieniu posiada standardowo otwarte 3 deskryptory plików:

  0 – stdin, 1 – stdout, 2 – stderr

Potoki/ Tworzenie potoków

POTOK to skierowanie standardowego wyjścia jednego procesu do standardowego wejścia

innego procesu. Operatorem tworzenia potoku jest pionowa kreska. Tworzenie potoku można przedstawić następująco:

proces | proces |… |proces

Tworzenie potoków można równocześnie łączyć z przekierowaniem wejścia i wyjścia, co pozwala na zaawansowane przetwarzanie strumieni danych.

proces | proces |… |proces > urządzenie wyjścia

Potoki mogą być także rozwidlone. Do rozwidlania potoku służy polecenie **tee**, które robi kopię strumienia danych i zapisuje ją w pliku, a oryginalny strumień przepuszcza dalej do strandardowego wyjścia lub kolejnego procesu.

Filtry

FILTRY to polecenia odczytujące dane z pliku lub strumienia, wykonujące na nich określone

działania i wysyłające wynik na standardowe wyjście.

- polecenie more, less – powodują wyświetlenie strumienia ekran po ekranie, more przewijanie do przodu, less przewijanie do przodu i do tyłu
- polecenie fold – powoduje wyświetlanie strumienia danych podzielonego na wiersze o podanej szerokości np. fold -5 lista.txt (wyśw. pliku lista.txt w wierszach o długości 5 znaków)
- polecenie fmt – powoduje wyświetlenie strumienia danych sformatowanego w wiersze o określonej szerokości BEZ DZIELENIA WYRAZÓW
- polecenie nl – numeruje wiersze strumienia danych „–n ln” wyrównanie do lewej strony „-n rn” wyrównanie do prawej strony „-i 2” skok co dwa „-v 5” początek num. od 5
- polecenie tee – kopiuje strandardowe wejście do pliku

- polecenie head – wyświetla zadaną liczbę pierwszych wierszy pliku, domyślnie 10
- polecenie tail – wyświetla zadaną liczbę ostatnich wierszy pliku, domyślnie 10
- polecenie wc – wyświetla informację o liczbie wierszy, słów i znaków znajdujących się w pliku (-l tylko wiersze, -w tylko słowa, -c tylko znaki)
- polecenie sort – sortuje zawartość strumienia (-r sortuje odwrotnie z do a)
- polecenie tac – wyświetla zawartość pliku wierszami od końca
- polecenie grep – wyszukuje wiersze zawierające ciągi znaków zgodne ze wzorcem (-E wyrażenia regularne, -i ignoruj wielkość znaków, -v szukaj gdzie nie ma wzorca)
- polecenie diff – porównuje każdy wiersz dwóch plików znak po znaku
- polecenie comm – porównuje dwa posortowane pliki wiersz po wierszu i wyświetla wynik w 3

kolumnach (1-tylko wiersze pierwszego pliku, 2 tylko wiersze drugiego pliku, 3 wspólne wiersze) (-1 -3 wyświetli tylko drugą kolumnę)

- polecenie sed – edytor strumieniowy (-n „3 p” wyświetl tylko 3 wiersz, „5 d” wyświetl bez 5 wiersza, „/abc/ d” bez wiersza zawierającego abc, sed „1,3 s/tak/nie/” zamiana w wierszach od 1 do 3 wszystkich wystąpień tak na nie
- polecenie cut – wybiera określone pola lub kolumny z pliku (-f1,3 –d”;” pobiera pole 1 i 3 oddzielone średnikiem, -c1-4 pobiera znaki od 1 do 4)
- polecenie tr – zamienia lub kasuje znaki w strumieniu danych (tr [a-z] [A-Z] = [:lower:][:upper:], abc 123
  tr –d –s „a” usuwa powtarzające się obok siebie litery a
- polecenie paste – łączy wiersze z różnych plików (-d „;” wstawi pomiędzy wierszami średnik)

- polecenie uniq – eliminuje powtarzające się wiersze wśród danych wejściowych (-c lista.txt wyświetla info ile razy dany wiersz występuje w pliku)
- polecenie split – dzieli zawartość pliku (strumienia danych) na porcje (-b liczbe bajtów, -l liczbe wierszy) i zapisuje je w odzielnych plikach
- polecenie hexdump – zamiernia zawartosc pliku (strumienia danych) na kody w systemie szesnastkowym
- polecenie sum, cksum, sha1sum, md5sum – tworzy sumy kontrolne. (sum = suma BSD, cksum = suma CRC, sha1sum = wartość funkcji haszującej SHA1, mh5sum = wartość f. haszującej MD5)
  Suma kontrolna to liczba uzyskana przez działania na danych, która służy do sprawdzania ich poprawności.

Wyrażenia regularne są to wzorce, które opisują ciągi/ łańcuchy znaków. Mogą określać zbiór

pasujących łańcuchów, mogą też wyszczególniać istotne części łańcucha.

Stanowią integralną część narzędzi systemowych takich jak sed, grep, edytorów np. vim, języków programowania przetwarzających tekst (np. awk, perl)

Polecenie grep z parametrem –E daje możliwość rozszerzonego interpretowania wyrażeń regularnych.

. znak dowolny
dowolny znak z klasy [[:alnum:]] – znaki alfanumeryczne, [[:alpha:]] litery, [[:digit:]] cyfry, [[:lower:]] małe litery [[:upper:]] duże
* występuje zero lub więcej razy
+ element występuje przynajmniej raz
? zero razy lub jeden raz

Konfiguracja powłoki

Do podstawowych elementów systemu operacyjnego Linux należy:

- powłoka (shell, interpreter poleceń) – program pełniący funkcję interfejsu pomiędzy użytkownikiem i jądrem systemu, interpretujący polecenia oraz umożliwiający uruchamianie programów, najczęściej wykorzystywaną powłoką w Linuxie jest Bash (Bourne Again SHell)
- jądro systemu – zbiór programów zarządzający procesami i zasobami systemu
- programy narzędziowe

Jednym ze sposób komunikacji pomiędzy wymienionymi elementami systemu operacyjnego jest wykorzystanie zmiennych powłoki.

Zmienne powłoki

Zmienna to cecha posiadająca NAZWĘ i przyjmująca pewną WARTOŚĆ. Polecenia wykorzystywane podczas pracy ze zmiennymi:

- set – wyświetlenie zmiennych powłoki tj *nazwa=wartosc (HOME=/home/nowakj)*
- echo – polecenie można wykorzystać do wyświetlenia wartości zmiennych, nazwę zmiennej należy poprzedzić znakiem dolara (echo uczelnia wyświetli uczelnia, ale echo $uczelnia wyświetli wartość np. UJ)
- operator „==” – zdefiniowanie nowej zmiennej lub zmiana wartości zmiennej już istniejącej (bez spacji wokół równości) (UCZELNIA==”UJ” stworzenie, UCZELNIA=”$UCZELNIA w Krk” dopisanie)
- pojedyncze apostrofy ‘ ‘ maskują nazwy zmiennych i sprawiają, że nioe pojawią się ich wartości (echo ‘Studiuje w $UCZELNIA da wynik dokladnie taki sam)

- podwójne cudzysłowy „ „ nie maskują zmiennych (echo „Studiuje w $UCZELNIA” da wynik studiuję w UJ)
- odwrotne cudzysłowy ` ` pozwalają zapisać wynik polecenia, to co umieścimy w cudzysłowach jest wykonywane (UCZELNIA=`echo 123` da wynik 123
- $() da efekt taki sam jak odwrotne cudzysłowy, tj. działanie wewnątrz zostanie wykonane, a jego wynik może zostać zapisany do zmiennej
- unset – usuwanie zmiennych powłoki

Przykładowe zmienne powłoki systemowej

Spośród wielu zmiennych powłoki, najczęściej wykorzystywane lub modyfikowane przez użytkowników są zmienne:

- PS1 – zmienna przechowuje definicję tzw. znaku zachęty (monitu systemu) (PS1=”Podaj polecenie:” będzie wyświetlać Podaj polecenie: Użytkownik definiując znak zachęty może w nim

zamieszczać specjalne kody, które są następnie zamieniane na odpowiednie wartości:

\! numer polecenia, \$ dla zwykłego użytkownika \# dla roota \d aktualna data \s nazwa powłoki \t aktualny czas \u nazwa użytkownika \w katalog bieżący \h nazwa komputera

- HOME – zmiany przechowuje ścieżkę dostępu do katalogu domowego użytkownika (echo $HOME wyświetli /home/nowakj
- PATH – zmienne przechowuje tzw. ściezkę poszukiwań. Dwukropek znajdujący się na końcu zmiennej PATH oznacza, że przeszukiwany będzie także katalog bieżący. W systemie Windows PATH najpierw przeszukuje katalog bieżący, potem reszte. W systemie Linux najpierw przeszukuje katalog zadany w PATH, potem katalog bieżący.

echo $PATH wyświetli /usr/bin:/usr/sbin a to przeszuka usr/bin i usr/sbin
echo $PATH wyświetli /usr/bin:/usr/sbin: a to przeszuka usr/bin, usr/sbin i kat.bieżący
NOWA ŚCIEŻKA DOSTĘPU DO ZMIENNEJ PATH:
PATH=$PATH:$HOME/programy :.

- SHELL – zmienna przechowuje nazwę programu będącego interpreterem poleceń (echo SHELL wyświetli /bin/bash)
- BASH – zmienna przechowuje ścieżkę dostępu i nazwę pliku zawierającego interpreter Bash
- BASH_VERSION – zmienna przechowuje informację o wersji powłoki bash (echo $BASH_VERSION wyświetli np. 4.2.45-release
- TERM – zmienna przechowuje typ terminala jaki jest wykorzystywany (echo $TERM wyświetli xterm)

Xterm jest to standardowy emulator terminala dla systemu X Window System

- HOSTNAME – zmienna przechowuje nazwę hosta (echo $HOSTNAME wyświetla Wizard)
- OSTYPE – zmienna przechowuje rodzaj systemu operacyjnego (echo $OSTYPE wyświetli linux-gnu)
- MACHTYPE – zmienna przechowuje opis systemu operacyjnego (echo $MACHTYPE wyśw x8..)
- USER – zmienna przechowuje nazwę użytkownika
- UID – zmienna przechowuje Unique User ID unikalny numer uzytkownika
- PWD – przechowuje ścieżkę dostępu do bieżącego katalogu
- HISTFILE – zmienna przechowuje nazwę pliku zawierającego historię wydanych poleceń
- HISTSIZE – zmienna przechowuje informację o liczbie wierszy przechowywanych w historii poleceń

- RANDOM – zmienna przechowuje wygenerowaną losowo liczbę całkowitą z przedziału 0 do 32767
- SECONDS – zmienna przechowuje liczbę sekund, które upłynęły od chwili uruchomienia powłoki. Jeżeli zostanie przypisana do niej jakaś wartość to przechowuje liczbę sekund od czasu przypisania oraz przypisaną wartość

Uruchamianie kolejnych powłok

Użytkownik w systemie Linux ma możliwość uruchamiania kolejnych powłok tzw. powłok potomnych. Służy do tego polecenie bash. Do zamknięcia otwartej powłoki potomnej służy polecenie exit.

Lokalne i globalne zmienne systemowe

Zmienna lokalna to zmienna dostępna tylko w bieżącej powłoce.

Zmienna globalna to zmienna widoczna w bieżącej powłoce i wszystkich powłokach

potomnych. Zmienną globalną tworzy sięza pomocą polecenia **export:**

- do tworzenia nowej zmiennej
- do wyeksportowania zmiennej już istniejącej

Zmienne stworzone w powłoce potomnej nie są widoczne z powłoce macierzystej.

Pliki startowe:

Użytkownik ma możliwość skonfigurowania systemu, w taki sposób, aby przy każdym logowaniu, uruchamianiu nowej powłoki lub kończeniu pracy system automatycznie wykonywał polecenia zawarte w tzw. PLIKACH STARTOWYCH. Mechanizm wykorzystuje się do:

- tworzenia lub modyfikowania zmiennych systemowych
- tworzenia nowych nazw poleceń (aliasów)
- uruchamiania lub zamykania programów

- wyświetlania komunikatów powitalnych lub pożegnalnych

  Wykorzystane pliki startowe to:

- bash_profile – polecenia zawarte w tym pliku są wykonywane każdorazowo przy logowaniu użytkownika
- bashrc – polecenia zawarte w tym pliku są wykonywane każdorazowo przy uruchamianiu nowej powłoki
- bash_logout – polecenia zawarte w tym pliku są wykonywane każdorazowo przy kończeniu pracy przez użytkownika

  Aliasy:

  Użytkownik ma możliwość definiowania nowych nazw dla poleceń tzw. aliasów. Służy do tego polecenie alias, a składnia to: alias nazwa=polecenie lub alias nazwa=’polecenie’

Polecenie alias bez żadnych argumentów i opcji wyświetla wszystkie zdefiniowane przez użytkownika aliasy.
Możliwość definiowania aliasów jest przydatna, kiedy użytkownik chce nadać np. łatwiejszą do zapamiętania nazwę jakiemuś poleceniu lub chce w skróconej postaci zapisać polecenie wraz z jego opcjami i argumentami. Do usuwania zdefiniowanych aliasów służy polecenie unalias.

ZMIENNE TABLICOWE

W powłoce bash można tworzyć jednowymiarowe zmienne tablicowe. Elementy tablicy są indeksowane liczbami całkowitymi (od zera). Indeksowanie nie musi być ciągłe.

- tablica=(wartosc1 wartosc2 wartosc3) utworzenie zmiennej tablicowej
- ${tablica[indeks]} odwołanie się do wartości elementu o podanym indeksie
  echo „Mój ulubiony kolor ${kolory[2]}

- ${tablica[*]} odwołanie isę do wszystkich elementów ze zmiennej tablica
- ${tablica[@]} to samo co wyżej
- tablica[indeks]=wartość dodanie elementu o wartości wartość i określonym przez nas indeksie
- ${#tablica[indeks]} odwołanie się do długości elementu o podanym indeksie ze zmiennej tablica
- ${#tablica[*]} odwołanie siędo liczby elementów ze zmiennej tablica
- ${#tablica[@]} to samo co wyżej
- unset tablica[indeks] usunięcie elementu o podanym indeksie ze zmiennej tablica
- unset tablica[*] lub unset tablica[@] lub unset tablica[] usunięcie zmiennej tablica

---

SKRYPTY POWŁOKI

Polecenia i struktury sterujące w skryptach:

- read – odczytanie wiersza ze standardowego wejścia i przypisanie go zmiennej

echo –n „Podaj swoje imie: „
read zmienna
podaje imie i zapisuje się w zmiennej $zmienna

- test lub […] porównanie wartości dwóch argumentów
- operatory porównań: -gt większe niż, -lt mniejsze niż, -ge większe lub równe –le mniejsze lub równe –eq równe –ne nierówność
  porównania napisów: -z testowanie pustego ciągu, -n testowanie wartości napisu, == identyczność napisów, != nieidentyczność napisów, str sprawdzanie czy ciąg znaków nie jest ciągiem pustym
  operatory logiczne: --a AND, –o OR, ! NOT
  operatory testowania plików: -f plik istnieje i jest zwyklym plikiem, -s plik nie jest pusty, -r możliwe jest odczytanie pliku, -w zapisanie pliku, -x wykonanie pliku, -d nazwa plik jest nazwa katalogu

- let – instrukcja służy do wykonywania działań matematycznych. mogą być wykorzystane następujące operatory matematyczne: * / + - % < > >= <= = (przypisanie), == (porównanie), != nierówność, & AND, | OR, ! Not
- $(()) wykonanie działań matematycznych
- if… then… fi //// if… then… else… fi – funkcja warunkowa
- until… do… done – pętla until działa dopóki polecenie testujące jest fałszywe
- while… do… done – pętla while działa dopóki polecenie testujące jest prawdziwe
- for zmienna in lista … do… done – pętla for wykonuje polecenia kolejno dla wartości podanych na liście
- for ((wyrażenie1; warunek; wyrażenie2)) do… done – pętla for wykonuje wyrażenie1, następnie sprawdza warunek, jeśli jest prawdziwy to wykonuje blok poleceń i oblicza wyrażenie 2

- continue – powoduje przejście do następnej iteracji pętli
- break – powoduje przerwanie wykonywania pętli
- case tekst in wzorzec1) … ;; wzorzec2) … ;; esac – struktura case dopasowuje wartość tekst do jednego z kilku wzorców. Jeśli wzorzec pasuje to wykonywane jest skojrzone z nim polecenie
- function nazwa_funkcji () {…return wartość} tworzenie funkcji

---

PRAWA DO KATALOGÓW I PLIKÓW

Podstawowe prawa do plików i katalogów

Właściciel zasobów może ograniczyć innym użytkownikom systemmu dostep do plików i katalogów dzięki możliwości nadawania i odbierania praw dostępu. Organizacja praw dostępu wygląda następująco:

- każdy plik i katalog posiada atrybuty określające prawo do:

- odczytania (r – reading)
- zapisania (w – writing)
- wykonania (x – executing)

- prawa dostępu określane są niezależnie dla:

- właściciela zasobu (u – user)
- członków grupy, do której należy właściciel zasobu (g – group)
- pozostałych użytkowników (o – others)
  wszyscy razem to a = all

Wyświetlanie praw dostępu to polecenie ls z opcją –l (-al)

1 znak to rodzaj zasobu:

- d – katalog
- - plik
- l – link symboliczny

- b – specjalny plik blokady
- c – specjalny plik znakowy
- p – potok
- s – gniazda

Link symboliczny wskazuje na inny plik lub katalog w tym samym lub innym systemie plików. Do tworzenia linków (dowiązań) służy polecenie ln –s cel nazwa_dowiązania

Link twardy wskazuje na pliki w tym samym systemie plików. Do tworzenia linków (dowiązań) służy polecenie ln cel nazwa_dowiązania

2-4 znak – prawa właściciela zasobu

5-7 znak – prawa grupy, do której należy właściciel

8- 10 znak – prawa pozostałych użytkowników

Od 2 do 10 znaku określa się mianem maski praw.

Do zmiany praw dostępu służy polecenie chmod, które można uruchamiać zarówno z maską liczbową, jak i prawami poszczególnych użytkowników (u, g, o, a) zapisanymi odpowiednimi symbolami (r,w,x)

Domyślne prawa dostępu

Domyślne prawa dostępu dla plików i katalogów są nadawane podczas ich tworzenia. Jeśli system nie korzysta z domyślnej maski praw to prawa wynoszą 666 dla plików i 777 dla katalogów.

Do zmiany maski praw służy polecenie unmask: zanegowana wartość binarna z jaką wywołane jest polecenie unmask jest poddawana koniunkcji (AND) z domyślną maską praw. Aby nadać domyślne prawa należy wywołać unmask 077.

Zmiana właściciela lub grupy pliku

Zmiany właściciela może dokonać tylko root lub użytkownik z jego uprawnieniami. Użytkownicy o

ograniczonych uprawnieniach mogą zmieniać grupę pliku, pod warunkiem, że do tej grupy należą.

Zmiana właściciela i grupy: chown login:grupa pliki

chown kowalski:pracownicy raport.doc zmieni właściciela na kowalski, a grupę na pracownicy
chown nowak lista.txt zmieni właściciela na nowak
chown :pracownicy lista txt chgrp pracownicy lista.txt <-zmieni grupę na pracownicy

chgrp –R pracownicy ./raporty zmieni grupę pliku rekursywnie dla całej zawartości raporty

Do sprawdzenia do jakich grup należy użytkownik należy polecenie groups.

Prawa specjalne

Poza podstawowym systemem praw Linux posiada także tzw. uprawnienia specjalne:

- SetUID/suid – ustaw identyfikator użytkownika, prawo pozwala uruchamiać pliki wykonywalne z uprawnieniami jego właściciela
  - oznacza to, że zwykły użytkownik może uruchomić program z prawami roota
  - znajduje to zastosowaniw w przypadku programów, które do prawidłowego działania wymagają wyższych uprawnień niż posiada użytkownik (np. zmiana hasła)
- SetGID/sgid – ustaw identyfikator grupy, prawo pozwala uruchamiać pliki wykonywalne z uprawnieniami grupy, do której one należą. Zasada działania jak SetGID
- sticky bit – bit lepkości/zaczepienia, nadane dla kataloga pozwala usuwać i zmieniać pliki oraz katalogi w nim zawarte tylko ich właścicielowi
  - chroni inne pliki i katalogi przed usunięciem lub modyfikację osoby, które nie jest ich właścicielem

- znajduje zastosowanie w przypadku katalogów współdzielonych np. /tmp

W dziwięcioznakowej definicji prawa specjalne nadpisują prawo wykonania (x) odpowiednio dla: właściciela, grupy, pozostałych. rwsrwsrwt

- SetUID i SetGID oznaczane są przez literę „s” lub „S” jeśli prawo wykonania dla użytkownika lub grupy nie jest nadane
- Sticky bit oznaczany jest przez literę „t” (na pozycji prawa wykonania pozostałych użytkow)

Nadawanie praw specjalnych realizowane jest za pomocą chmod z wykorzystaniem

- operatora + i – oraz symboli „s” „t” = u+s SetUID, g+s SetGID, +t stickybit
- liczbowej maski praw, w której dodatkowe cyfry oznaczają:
6 SetUID i SetGID, 4 SetUID, 2 SetGID, 1 sticky bit

ZARZĄDZANIE PROCESAMI

Charakterystyka procesów

Proces jest to program realizowany przez system operacyjny. Wyróżniamy:

- procesy użytkownika – programy i polecenia uruchomione przez użytkownika
- procesy systemowe – programy uruchomione przez system operacyjny

Systemy operacyjny umożliwiające współbieżną realizację procesów to:

- systemy wielozadaniowe **–** w tym samym czasie system operacyjny realizuje więcej niż jedno zadanie
- systemy z podziałem czasu – czas pracy procesora dzielony jest pomiędzy poszczególne procesy

Przeciwieństwem są systemy jednozadaniowe np. MS DOS

Każdy proces charakteryzuje siępewnymi atrybutami:

- przestrzeń adresowa
- licznik programu
- licznik stanu
- licznik rejestru
- deskryptory pliku
- dane procesu
- zależności rodzinne (proces macierzysty/ proces potomny)

Jądro systemu może sterować procesem i ustawić go w kilku stanach:

- pracujący w trybie UŻYTKOWNIKA (kod procesu wykonywany przez procesor)
- pracują w trybie JĄDRA (jądro wykonuje wywołanie systemowe wykonane przez procesor)
- uśpiony (proces czeka na jakieś zdarzenie np. odczyt danych z dysku lub otrzymanie danych z sieci)

- gotowy do wykonania (można uruchomić w każdej chwili, ale nie ma jeszcze przydzielonego procesora)
- zombie (proces zakończył działanie i czeka na odebranie kodu powrotu przez proces macierzysty)

Procesy w systemie Linux

Proces otrzymuje dostęp do zasobów takich jak procesor, pamięć, pliki, urządzenia wej-wyj. Do wyświetlania dokładnych informacji o procesach służy polecenie **top / htop**

Wyświetlane kolumny: PR priority, NI nice value, VIRT virtual memory size, RES resident memory size, SHR shared memory size, S proces status, TIME+ Cpu time, COMMAND command line

Każdy proces posiada swój unikalny identyfikator liczbowy PID oraz identyfikator rodzica PPID

(Parent Process ID) czyli procesu, który go utworzył. Numer pid procesu można uzyskać poleceniami **pidof lub pgrep**

pgrep –u kowalski –l „^a” = wyświetli wszystkie PID procesów użytkownika kowalski z podaniem ich nazw (-l) zaczynające się na a

Strukturę uruchomionych procesów można przedstawić w formie drzewa, w którego korzeniu znajdować się będzie proces z PID równym 1 czyli init lub systemd, uruchamiany jako pierwszy po załadowaniu jądra. Do wyświetlenia drzewa procesów służy polecenie **pstree**.

pstree –p = wyświetli drzewo procesów wraz z ich PID

Procesy posiadają swoich właścicieli zazwyczaj są to użytkownicy, którzy uruchomili dany proces, za wyjątkiem programów z ustawionym bitem SetUID lub SetGID. Prawa dostępu do zasobów

systemu (np. katalogu /dev) użytkownika, który uruchomił proces określają prawa dostępu jakie będzie posiadał proces.

Procesy jako programy mogą znajdować się w określonym stanie:

- R – proces działający (running)
- T – proces śledzony lub zatrzymany przez sygnał (traced)
- D – proces uśpiony w stanie nieprzerywalnym, OCZEKUJE na dane z urządzenia wej/wyjścia (disk wait)
- S – proces uśpiony (sleeping)
- Z – proces „duch” (zombie)
- X – proces martwy (nie powinien być wyświetlany)

Sygnały procesów

Procesy komunikują się z jądrem systemu oraz pomiędzy sobą poprzez mechanizmy komunikacji IPC (Inter-Process Communication Mechanism). Należy do nich komunikacja z użyciem sygnałów, które są przerwaniami programowymi.

Sygnały mogą być generowane przez:

- jądro systemu
- procesy
- użytkownika z wykorzyst. polecenia lub funkcją kill
- użytkownika z użyciem kombinacji klawiszy

Proces może komunikować się, czyli wysłać sygnał innemu procesowi, jeśli oba posiadają tego samego właściciela oraz te samą grupę (to samo uid i gid). Jądro i root mogą wysyłać sygnały bez ograniczeń. Sygnałów nie odbiera tylko init(PID=1)

Sygnały dzielą się na:

- maskowalne, czyli te, które proces może obsłużyć (zablokować, przechwycić, zignorować)
- niemaskowalne, czyli te, których nie można obsłużyć SIGSTOP SIGKILL

Najpopularniejsze sygnały:

- SIGSTOP (kod 19) zatrzymanie procesu, sygnał niemaskowalny. Uzytkownik może wywołać go przez ctrl + z
- SIGTERM (kod 15) programowe zakończenie procesu, pozwala zamknąć otwarte pliki i zwolnić pamięć. Sygnał standardowy, generowany domyślnie przez kill, do wyłączania systemu.
- SIGKILL (kod 9) zakończenie procesu, sygnał niemaskowalny. Pewne zakończenie procesu
- SIGQUIT (kod 3) zakończenie procesu przez użytkownika przy pomocy terminala. kończy pracęz zapisem obrazu pamięci, użytkownik może go wywołać przez ctrl + [\]

- SIGINT (kod 2) przenoszenie procesu, domyślnie zamyka proces. Uzytkownik może wywołać go przez ctrl + c
- SIGHUP (kod 1) zerwanie łączności, sygnał zawieszenia. wysyłany przy zerwaniu łaczności z terminalem do wszystkich procesów, dla których jest terminalem sterującym

  Polecenie **kill** służy do wywołania sygnałów procesom o podanym numerze zadania lub numerze PID oraz do wyświetlania informacji o dopuszczalnych sygnałach. kill –l wyświetla listę dopuszczalnych sygnałów.

  Procesowi można wysłąć sygnał używając jego nazwy lub korzystając z jego PID:

  kill –SIGKILL 2345 = kill -9 2345 zabije proces o PID 2345

Polecenie **killall** pozwala na przesyłanie sygnału wykorzystując nazwę procesu lub wyrażeń regularnych.

killall –i –v vim =zabije wszystkie procesy o nazwie vim, -i zapyta o zgodę na zabicie, -v poinformuje o udanym wysyle sygnalu

killall -9 -1 = -9 sigkill czyli zabicie, -1 czyli wszystkich procesów o wartości -1

Jeśli nie ma nazwy lub kodu sygnału to wysyłany jest SIGTERM.

Priorytety procesów

Priorytet prcoesów określa liczba nice (domyślnie 0), może przyjmować wartości od -20 do +19.

Uruchomienie procesu z wyższym priorytetem lub zmiana priorytetu na wyższy może być przydatna, gdy użytkownik chce przyspieszyć wykonywanie jakiegoś zadania np. kopiowanie lub kompresja pliku.

Zmiana priorytetu na niższy może być przydatna, gdy system jest zbyt obciążony przez jakis proces (np. maszynę wirutalną), a użytkownik chce dokonać szybciej inne zadania.

Do uruchomienia procesu z podanym priorytetem służy komenda nice –n priorytet polecenie.

Do zmiany już uruchomionego procesu służy komenda renice –n priorytet -p PID

Zarządzanie procesami użytkownika w systemie Linux

Nowe zadanie użytkownika np. program, polecenie, uruchamiane jest domyślnie na pierwszym planie, a pozostałe zadania są w tle. Użytkownik może zarządzać zadaniami, może:

- uruchomić zadanie w tle poprzez znak **&**
- zatrzymać proces na pierwszym planie i przenieść go w tło poprzez **ctrl + z**

- wyświetlić informacje o stanie zadań w tle poprzez polecenie **jobs**
- wyświetlić informację o procesach systemowych poprzez polecenie **ps**
- przesunąć zadanie z tła na pierwszy plan poprzez polecenie **fg** +/-, PID, nazwą procesu
- uruchomić zatrzymane w tle zadanie poprzez polecenie **bg**
- zakończyc wykonywane przez czasem zadanie:
  - pierwszoplanowe: ctrl +c
  - zadanie w tle: kill z numerem procesu systemowego, %numerem zadania lub %+ / %-

Bezpośrednia komunikacja z użytkownikami

Użytkownik może komunikować się bezpośrednio z inynmi użytkownikami za pomocą:

- talk nowakj (aby przyjąć rozmowe ktoś musi odpisać talk nazwiskozaczynajacegorozmowe np.

talk kowalkip, aby odrzucić należy nacisnąć ctrl +c)

- write login, ctrl+d
- przyjmowanie wiadomości mesg y, odrzucanie mesg n

DOS

Rozwój trybu tekstowego

Bezpośrednim poprzednikiem trybu tekstowego w systemach MS Windows był system operacyjny DOS (Disk Operating System) z interpreterem poleceń COMMAND COM.

System operacyjny DOS na przestrzeni lat był produkowany przez różne firmy i występował w różnych wersjach np.

- QDOS – pierwowzór napisany przez Tima Patersona i wykupiony przez firmę Microsoft
- MS-DOS – firmy Microsoft

- PC-DOS – firmy IBM
- DR-DOS - firmy Digital Research
- Free-DOS - produkt tzw. wolnego oprogramowania

Pierwsza wersja systemu MS-DOS 1.1 została opublikowana w 1982, ostatnia samodzielna wersja MS-DOS 6.22 została opublikowana w 1994. Późniejsze wersje systemu stanowiły integralną część systemu Windows:

- MS-DOS 7.0 w Windows 59
- MS-DOS 7.1 w Windows 95 OSR2, Windows 98 i Windows 98SE
- MS-DOS 8.0 w Windows ME w roku 2000 (ostatnia wersja DOS)

PODSTAWOWE PLIKI SYSTEMU MS-DOS:

- IO.SYS, MSDOS.SYS - pliki zawierające jadro systemu operacyjnego

- COMMAND.COM – interpreter poleceń wewnętrznych
- CONFIG.SYS – plik konfiguracyjny systemu służy m.in. do zarządzania pamięcią, ładowania programów rezydentnych i sterowników urządzeń
- AUTOEXEC.BAT – plik wsadowy uruchamiany po przetworzeniu pliki CONFIG.BAT, służy m.in. do ustawiania zmiennych systemowych, ładowania sterowników, definiowania znaku zachęty.

NAZEWNICTWO PLIKÓW I KATALOGÓW

System 8+3, w nazwie i rozszerzeniu nie mogą być używane znaki o kodzie ASCII mniejszym lub równym 32 ani znaki takie jak . , „ \ / [ ] : | < > + = ;

Dla programów w postaci binarnej zarezerwowane są rozszerzenia EXE i BAT. Różnica pomiędzy nimi jest taka, że BAT to zazwyczaj mniejsze pliki niż EXE, BAT pracują na jednym segmencie kodu, EXE na wielu.

NAPĘDY, ŚCIEŻKI, URZĄDZENIA

- napędy A: C:
- katalog główny \
- katalog bieżący .
- katalog nadrzędny ..
- ścieżka bezwzględna D:\uek\so\uczelnia\plik.txt
- ścieżka względna ..\so\uczelnia\plik.txt
- urządzenia puste NUL lub NUL:
- drukarki PRN, PRN:, LPT1, LPT2
- port szeregowy AUX, COM1, COM2

Polecenia systemowe dzielą sięna wewnętrzne oraz zewnętrzne. Wprowadzanie poleceń odbywa się poprzez wprowadzanie odpowiednich komend w trybie tekstowym.

Parametry poleceń podaje się po znaku /, a pomoc dostępna jest z poleceniem /?

POLECENIA SYSTEMOWE

- attrib – wyświetla lub zmienia atrybuty plików

- cd (chdir) – wyświetla nazwę bieżącego katalogu lub zmienia go
- chkdsk – skanowanie dysku
- cls – czyści ekran
- choice – umożliwia wybranie jednego elementu z listy wyboru i zwraca indeks zaznaczonego elementu
- copy con ctrl+z/ F6 - kopiuje konsolę do pliku
- copy - kopiuje jeden lub więcej plików do podanej lokalizacji
- date – wyświetla i ustawia datę
- del/erase – usuwa pliki
- deltree – usuwa katalog z zawartością (niedostępny)
- dir /p /w – wyświetla zawartość katalogu, p-wstrzymuje wyświetlanie po zapełnieniu ekranu, w-stosuje format szerokiej listy
- diskcopy a: a: - kopia dysku na drugi

- echo – wyświetla komunikaty lub włącza/wyłącza wyświetlanie poleceń
- @echo – wyświetla stan echo
- echo on|off - włącza wyłącza echo
- fc – porównuje zawartość dwóch plików lub zestawów plików i wyświetla różnice między nimi
- format /q /s – formatuje dysk, q-szybki format, s-tworzy dysk
- label – wyświetla etykietę dysku
- mk/mkdir – tworzy nowy katalog
- mem /c /p – sprawdza stan pamięci konwencjonalnej /c porządkuje programy w pamięci konw. oraz w upper memory /p porządk. programy wraz z adresem i ich rozmiarem
- move – przenosi pliki i zamienia nazwy plików i katalogów
- pause – zawiesza przetwarzanie pliku wsadowego i wyświetla komuniakt ‘aby kontynuowac naciśnij dowolny klawisz’

- rd /s /q – usuwa katalog, s-z zawartością, q-quiet, bez pytania
- ren – zmiana nazwy pliku
- set – wyświetlenie, usuwanie i zmienianie zmiennych środowiskowych programu cmd
- subst – kojarzy ścieżkę z literą dysku, napowanie katalogu
- sys c: a: - przenosi napęd
- time – wyświetla i ustawia czas
- type – wyświetla zawartość jednego lub więcej plików tekstowych
- undelete , unformat
- ver – wyświetla wersję Windows
- verify on|off – włącza przeprowadzanie przez program cmd weryfikacji poprawności zapisywania danych na dysku
- vol – wyświetla etykietę woluminu dysku i numer seryjny, jeśli istnieją
- xcopy – kopiuj katalog z zawartością

ZMIENNE SYSTEMOWE:

- definicja zmiennych systemowych: set zmienna=nazwa
- odwołanie sięgo wartości zmiennych %nazwa% np.
  set PATH=C:\NC;%PATH%
- wybrane zmienne systemowe:
  - PATH przeszukiwanie,
  - COMSPEC ścieżka wskazująca interpreter poleceń
  - TEMP, TMP zmienna określa położenie folderu na pliki tymczasowe
  - PROMPT określa wygląd znaku zachęty dosa

ATRYBUTY PLIKÓW

- S – systemowy
- H – hidden, ukryty
- R – read only, do odczytu
- A – archiwalny

Nadawanie atrybutów np. attrib +a –r plik.txt

Po nadaniu praw system wyświetli je:

- S – nie można usunąć
- R – nie można dopisać ani usunąć
- H – nie widać pliku

STRUMIENIE DANYCH I POLECENIA FILTRUJĄCE:

- operatory przekierowania strumienia danych dir > plik, dir >> plik // echo tekst > plik // type plik > prn // type plik >> plik2
- filtry more (dzieli na strony), sort (sortuje) find /i (wyszkuje z pominięciem znaków)

PLIKI WSADOWE są to pliki tekstowe zawierające zestaw poleceń systemowych dla interpretera poleceń, wykonują instrukcje wiersz po wierszu i mają rozszerzenie .bat

PRZYDATNE POLECENIA

- kometarz w pliku wsadowym to rem lub ::
- instrukcja skoku goto etykieta
- poziom błędu errorlevel wraz z instrukcja if

if errorlevel=x goto etykieta

- instrukcja choice
  choice /c:TN /t:T,5 Co mam nacisnąć
- instrukcja if [not] exist plik
- odwołania do parametrów %0 nazwa, %1 parametr pierwszy
- instrukcja przesunięcia parametrów w lewo SHIFT, pozwala na użycie więcej niż 9 parametrów
- wykonanie polecenia dla wszystkich elementów z listy
  FOR %zmienna IN (lista) DO polecenie
  zmienna musi być jednoliterowa, %zmienna jest na poziomie wiersza poleceń, %%zmienna w plikach wsadowych

---

LOSOWE POLECENIA

Wyświetl z lista.txt wiersze od 20 do 25:

cat lista.txt | sed –n „20,25 p”

Wyświetli z pliku lista.txt te wiersze, które zaczynają się od wielkiej litery A, B lub C, nie zawierają cyfr oraz nie mają więcej niż 30 znaków.
cat lista.txt |grep –E “[^0-9]” |grep –E “^[A-C] .{0-30}$”

Usunie z pliku lista.txt wszystkie kropki i przecinki.

cat lista.txt |tr –d „.,” > bufor.txt
cat bufor.txt > tr.txt |rm bufor.txt

Jakie użytkownik musi mieć prawa do pliku i katalogu, w którym ten plik się znajduje, aby mógł uruchomić ten plik?
dla katalogu minimalnie --x, dla pliku minimalnie r-x

Utworzy katalogi z nazwami zalogowanych użytkowników:

```
for x in $(users)
do
mkdir $x
done
```