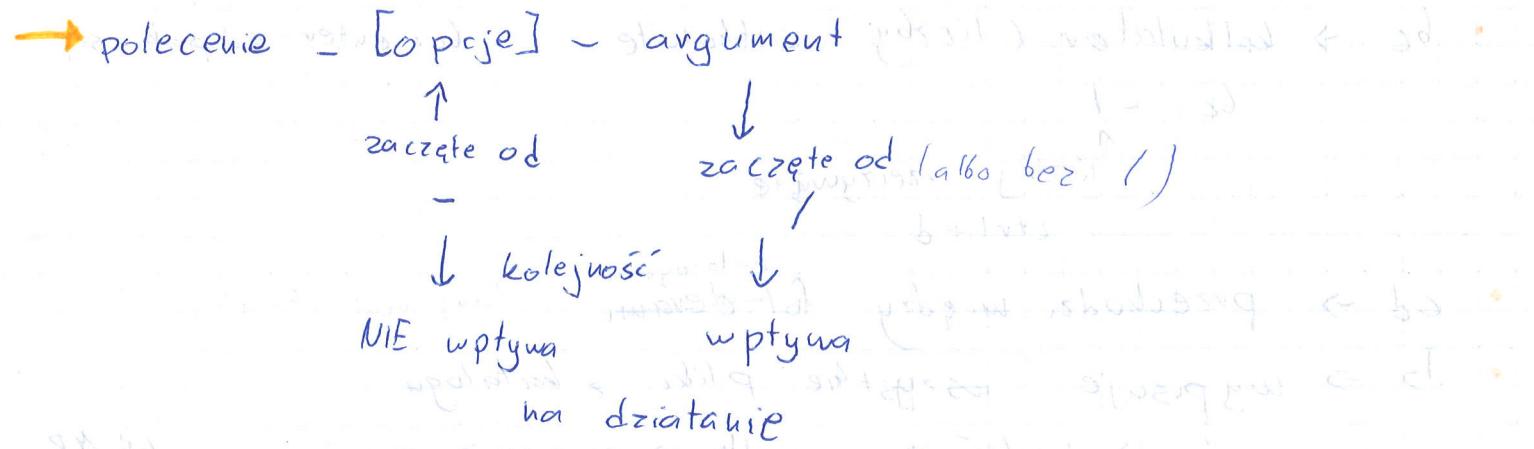


UNIX - laborki nr 1 (dla zalogowanych)



Polecenia

- pwd → wypisz katalog domowy (! scierka do aktualnego katalogu)
- file → sprawdza format pliku
 - argument = nazwa pliku
 - jeśli wypisze data - nie zna tego formatu
- cat → wypisuje plik
- more → pokazuje dalsze kawałki pliku (jeśli ujemnie się na ekranie to wyświetla tylko spacje - przechodzi się na koniec strony, następny dopiero Q - opuszcza się)
- less → pozwala powtarzać się strzałkami, pg up, pg down
 - argument wyszukuje dane stowarzyszone z " - przesunięcie
- head → wypisuje 10 pierwszych linii pliku
 - opcje → liczba linii
 - argument → nazwa pliku
- tail → 10 linii od końca (analogicznie)
- wc → wypisuję liczbę: 50 111 277 (liczba znaków w pliku, zawsze ta liczba stowarzyszona z kolejnością)
- kiedy wypisać tylko wybrane: -l -w \$ -c (opcje)

• |
 ↑
 znak specjalny/
 więc nie musi być
 spacji

 wyższy czytać z wejścia

 to musi wypisywać na wyjście

up. head -17 /etc/passwd | tail -1

- expr → kiepski kalkulator
 - bc → kalkulator (liczby dziesiętne) ↗ bc + enter - od pak. się
 - bc -l
liczby rzeczywiste
ctrl+d
 - cd → przechodzi między foldery (argument = katalog)
 - ls → wypisuje wszystkie pliki z katalogu
 - l → w liscie ↗ ls -l ↗ wypisuje rozmiar up. w kB, MB
human readable (czytaj)
 - la → otwiera ukryte pliki
- użytkownik grupa inni xz. 1 root root 1,70 K rozmiar locale.h
 prawa dostępu d - dostęp do wszystkiego modyfikacji

Nie używać spacji w nazwach plików! kropka, podtak, podtak

- katalog bieżący
- katalog nadwzędny
- history → wypisuje historię wykonywanych poleceń

ściezka względna - mówi o tym, jak dojść do danego pliku względem bieżącego katalogu

bez względnej ścieżki zaczyna się od /

cat (brak argumentu) + enter



powstaje program

std::in std::out std::err

0 1 2

wjście wyjście wyjście

diagnostyczne

Cokolwiek wpisujesz + enter → wypisuje to

jak zakończyć proces?

→ ctrl+d

albo

→ pisz coś + enter + ctrl+c

UNIX-laborki nr 2

• **touch** → tworzy plik (pusty)

argument = nazwa jednego lub wielu plików

- **rm** → usuwa plik (remove)
argument jeśli plik nie ma w, to zaproś się has czy go usuwać. y/eut/e → usuwanie
- **mkdir** → tworzy katalog
argument = nazwa
- **rmdir** → usuwa katalog

! jeśli w katalogu coś jest, nie można go tak usuwać
-r (usuwanie rekurencyjne - wszystko od do tego miejsca w drzewie katalogowym)
rm -r katalog

• **cp** → kopiuj

argumenty = nazwy plików i miejsce docelowe kopiowania

(można więcej niż jeden plik) • - katalog bierzący

-r (wszystko odtąd w dół drzewa katalogów)

• **mv** → move (zmienia nazwę), lub przekształca nazwę

argumenty = obecna nazwa i nowa nazwa (może być więcej kierunkiem specjalnym) w cudzostwach! "14:" dosunięte do tegoż argumentu

grep [opcja] wyrażenie [plik] argument

• **cut** → wycina konkretne linie i kolumny

cut -d: -f 3,5 cat /etc/passwd | cut -c3,5-8
delimited kolumna 3,5

• **uniq** → jeśli kilka identycznych linii następuje po sobie, to wyświetla tylko jedną taką linię

• **sort** → sortowanie

-n (number) traktuj to jako liczby sort -un

• **who** → kto jest zalogowany pozwala wejście, godzina logowania

• **w** → kto jest zalogowany, jakich procesów używa i dużo innych ip IDLE - ile czasu minęło od ostatniego polecenia, JCPU - czas używany przez wszystkie procesy,

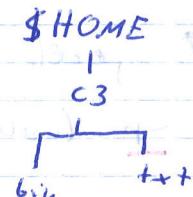
PC PV - czas wygrywaj paze aktualny proces, historia awarii, podgladem do systemu

- last → ~~czyta od końca do poczytka ???~~ podaje informacje o logowaniach użytkownika do systemu
 - awk → trochę jak cut
 - F ustala nowy separator kolumn

```
up: cat /etc/passwd | awk -F: '{print $3" "$5}'  
{if ($3==69) print $5}'
```

Laborki 3

Jak utworzyć drzewo katalogów?



- mkdir c3 c3/txt c3/bin

rm !-rfc 3 (kasuje wszystko w katalogu c.3)

- `mkdir -p 3/6iu c3/txt`

funkcja -p tworzy katalog rodzica, jeśli nie istnieje

Polecenia:

→ polecenie > plik - zapisz w pliku to, co polecenie wygeneruje na standardowe wyjście. Usuwa stan za pomocą pliku, tworzy plik, jeśli nie istnieje

→ poleceń >> plik - dopisuje do obecnej zawartości pliku lub tworzy nowy plik

→ man → informacja o tym, co robi dane polecenie

→ diff → wskazuje roznice między plikami

(204 d 203 - rozulica między liniami 204 a 203)
plik 1. plik 2.

→ chmod → zmiana praw dostępu

chmod [opcje] zmiana [plik]

- sposób "względny" - dwobuatowy filtracji

Koum co robic prava ulgololga + - r/w/x/s/l

- albo wszystkie prawa na nas (sumowanie)

u.p. 5 79? 7658

Laborki 4

Okna tekstowe przy logowaniu \rightarrow $ctrl+alt+F(1/2, \dots)$

Polecenia: [Napisy] \rightarrow [System] -> [Zmiany systemu]

- quota ->

wypisuje limity na pliki i bloki dla danego użytkownika

blocks quota limits grace
(limit (limit) ile zostało (np. 7 days))
miękkie) (twardy)

files quota limits grace

- dd (device to device) \rightarrow kopiowanie bloków z urządzenia do urządzenia

katalog /dev

- /dev/urandom - generator nieskończonej ilości bajtów o wartościach pseudolosowych
- /dev/zero - generuje zera
- /dev/null - wszystko zgada :D

bs - block size count=60000 (ile się wygeneruje)

- df - stopień zajęcia dysku

1 kolumna: ścieżka dostępu do systemu plików

informacje o blokach danych, gdzie ten blok jest zamontowany

-o (output)

Kontynuowanie procesu zarządzanie procesami

- $ctrl+z$ \rightarrow zatrzymuje proces
- jobs \rightarrow wypisuje procesy (działające w tle) \uparrow nr procesu
- bg \rightarrow wywołuje proces w tle (bg %1) - bez parametru najświejszy
- fg \rightarrow wywołuje proces "na wierzchu" (fg %3)
- ps \rightarrow wypisuje info o procesach z bieżącego terminalu

PR 1 - wykorzystanie procesora

PS -ie (rózne procesy) ← unikalny typ działań

- zaatrzymanie procesu (usunięcie) → kill [opcja] identyfikator
SIGTERM
 - SIGINT = ctrl + C kill g - usuwa proces
 - nice → zmiana wartości parametru wykorzystania procesora
im wyższa wartość tym mniej pracy dla procesu
 - renice [+/- wartość] identyfikator
 - killall

\$\$ - zwieburg

size

RSS: newsreader - [RSS](#)

RSS = pamięć rezy dualna

— 65230

Shift \hookrightarrow Precessua argumenty (w/ perf.)

Laborka 5

echo → wypisywanie tekstu na wyjście

np. echo "ściezka dostępu: \$PWD"

(-n - nie przechodzi do nowej linii)

set → ustawia zmienne parametryczne (\$1)

(lub wyświetla wartości zmienne)

env → ustawia zmienne środowiskowe

(lub wyświetla zm. środow.)

alias → tworzenie aliasu poleceń: ~~alias~~ alias [nazwa]='[definicja]'

echo \$PATH (wypisuje zmieniącego się PATH)

zmienne - zapisuje się dużymi literami

sh → interpreter virtualny (exit)

jeżeli utwarzymy zmianę, to jest ona przenoszona tylko do set, nie do env, więc nie istnieje w innym środowisku (utworzonym przez sh)

export → przekosić zmianę do env (eksportuje do innych terminali)

* PATH - wybierze to polecenie (przy wyszukiwaniu) zarówno dotyczące do poleceń systemowego o tej samej nazwie

alias → alias nowa_nazwa="comanda"

which → wypisuje ścieżkę do wybranego pliku ~~z wykonywaniem~~

shift → przesuwa parametry ^{which gedit}

unalias [nazwa] → usuwa alias

Znaki:

→ ` ` zabezpiecza przed interpretacją

→ ' ' nie jest w ogóle interpretowany

→ " " interpretowany, ale odporny na problem ze spacja, zastąpiony wynikiem wykonania polecenia między akcentami

* dopasowuje pliki w bieżącym katalogu (np. wypisz-pliki) *atut

Włocławek bawarski [a, 6] {1, 2}

Laborka 6 - skrypty z p

pętla: \$while [zakres]; do

LP = 'jakiś polecenie'

done

test →

czy if, test:

-a (konieczna)

-o (alternatywa)

! - negacja

wi → przejście do skryptu (np. vi skrypt)

wget →ściąga plik

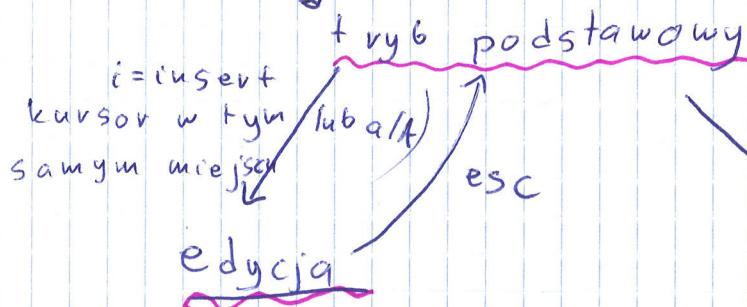
tar → rozpakowuje plik (xzvf)

if [-z "\$"] - zwraca prawdę, kiedy jest pusty

skip →

noerror →

skrypt2



a - przesuwa jedną literę w prawo

A - przesuwa na koniec linii

w - zapisuje do pliku (w nazwa-pliku)

wq - write, quit

q - quit

q! - wyjście bez zapisu

1. skrypt4 - wykonuje skrypt

#!/bin/bash - początek skryptu

normalne # - oznacza komentarze aż do końca linii

Zmienne

→ programowe

opcje: r(tylko do odczytu), i (liczba całkowita), a (tablica)

f - wyświetla ciało funkcji

→ specjalne (po \$)

0 - nazwa skryptu ? - wartość zwracana przez ostatnie polecenie

1-9 - nazwy argumentów \$ - PID bieżącej pątaki

@ wszystkie argumenty # - ilość argumentów

→ środowiskowe (np. HOME, PWD, OLDPWD, USER)

trap → przechwytuje sygnały (wyświetla na początku skryptu)

trap 'echo "costam"; exit 1' SIGINT

(jeśli nie ma tu żadnego polecenia blokuje)

misja ob strony

sceny

Skrypty c.d.

→ Tablice

tablica=(9 3 2 "diag znaków")

echo \${tablica[2]}

echo \${tablica[*]} - wypisuje całą tablicę, może być teraz @

echo \${#tablica[3]} - ile znaków ma element trzeci

echo \${#tablica[@]} - ile elementów ma tablica

→ Operacje matematyczne

a=1

echo a=\$[a+10] lub let a=a+10 (let a+=10)

** - potegowanie

expr \$[5*9]

→ Instrukcje warunkowe

- if

* (-lt (<), -le (<=), -gt (>), -ge (>=), =, !=)

* inne operatory:

→ a - plik istnieje i jest:

→ n - wyrażenie ma dtugosć > 0

• b - blokowym plikiem specjalnym

• z - ma zerową dtugosć

• - plik znakowy

• distruje i jest katalogiem

• h - linkiem symbolicznym

→ P1 -ut P2 (P1 mawiący o dP2)

• f - plikiem zwykłym

• P1 -ot P2 (P1 stawiający dP2)

→ rwx - nowa & czytać, zap, wykonywać

→ N-plik istnieje i był zmieniany od

ostatniego uruchomienia

if []

then
else)

exit=return

fi

if [\$!losc="1"]

then

elif [\$!losc="2"] else if

then

(*)) - wzorzec domyślny
inny pasuje do innych ifów

read → wczytaj z wejścia

• case (np. case "\$miesiąc" in "1") echo "Styczeń";;

read - a [nazwa-tablicy] - wpisuje do tablicy kolejne
wyrazy do kolejnych indeksów

zmienna REPLY - chwilowa zmieniona zapisująca z read

→ Pętle

- for ~~while~~

- i in {0...6}

- i in `seq 0 \${~~rozmiar~~ rozmiar-1}`

seq → wypisuje wszystkie liczby z podanego zakresu,
rozmiar = \${# tablica[@]}

seq -w

tr → zamienia wielkość znaków

- while

while [do] done

- until (odwrotność while - dopóki warunek jest ta sama skladnia)
- select - raczej nie będzie potrzebny

→ Funkcje

zmienna \$g globalna (chyba reuzujemy local), nie ma return, trzeba funkcji drukuj { nowaFunkcja () { zapisac do zmiennej i dac echo }

}

}

- z argumentami

function Funkcja {

arg1=\$1

arg2=\$2

wywołanie: Funkcja "argument1" "argument2"

other function

z dotyczeniem innego pliku z funkcjami:

• 1 functions (funkcje)
• katalog
• ten sam katalog

passwd → zmienia hasło

prawo do zmiany hasła - użytkownik ma prawo

id → wyświetla informacje o użytkowniku

id - s.

-u podaje tylko identyfikator

-g tylko grupę efektu

-G wszystkie grupy

nice [-n wartość] polecenie → wykonuje polecenie z podanym priorytetem

Proces:

- identyfikator (PID)
- proces macierzysty, który go uruchomił (PPID)
- identyfikator grupowania procesów (PGID)
- NICE - ustalanie priorytetu (-20 - 19)
- wstawienie grupy w root
- terminal (ctrl + Z)

Różne rozkazy

cat

clear

!! - zwraca ostatnie polecenie

!n - zwraca polecenie o danym numerze w historii

! - zwraca polecenie wywołane dany raz polecem tym

!tekst - zwraca ostatnie polecenie zaczynające się od "tekst"

\$? - zmienia zawierającą kod zakončenia ostatniego polecenia

nano - prosty edytor tekstu

stat - informacje pliku i systemu plików

Prawa dostępu - zapis cyfrowy

otoczki ninsins ← b w eę nq

• 1 - SUID

pliki - powoduje wykonanie zawsze z uprawnieniami właściciela katalogi - dziedziczenie właściciela w nowych plikach

• 2 - SGID

pliki - wykonanie zawsze z uprawnieniami ich grupy

katalogi - dziedziczenie grupy

• 3 - sticky bit

pliki wykonywalne - powoduje zachowanie uprawnień dla

szysiego dostępu - uprzejmosten - 3D1U

katalogi - usuwanie i zmiana nazwy zawartości tylko dla

kolejne cyfry: 4 - odczyt, 2 - zapis / zmiana zawartości, 1 - wykonanie (odczyt serwera)

Zapis literowy

• Pierwsza litera - typ pliku

- b, c - użycie blokowe/znakowe

- d - katalog

- f - zwykły plik

- l - link (dowiązanie symboliczne)

• Prawa specjalne (zamiast x)

s - w 1. formie SUID z wykonaniem, w 2. formie SGID z wykonaniem

o - odczyt - nie - wykonywanie

z - sticky bit z wykonaniem (3. forma)

T - 71 - bez wykonywania powinno być puste

/etc/passwd

nazwa: x: UID(użytk): GID(grup podst): opis: /home: interpreter

/etc/group

nazwa: x: identyfikator: członkowie