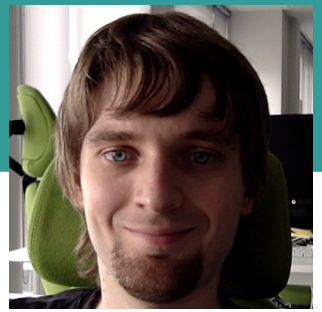
LINUX



Cześć

Maciej Kosiedowski

Software Engineer @ Schibsted Tech Polska

Linux registered user #325402 since 2003

Organizacyjnie

19.11.2018

- Pierwsza część – 3h
- Przerwy na żądanie
- Wprowadzenie do Linuxa
- Jak poruszać się w systemie

28.11.2018

- Druga część – 3h
- Instalacja programów
- Programowanie w Bash

Prezentacja i rozwiązania ćwiczeń dostępne w:
https://github.com/infoshareacademy/jjdd5-materialy-linux

Co to jest Linux?

Linux to (najczęściej) darmowy system operacyjny, tworzony przez społeczność, zgodny ze standardami POSIX, łatwy do dostosowania do swoich potrzeb

Co to POSIX?

To standaryzacja systemu Unix

Opisuje interfejs programistyczny, interfejs użytkownika i właściwości powłoki systemowej

Przykłady systemów: macOS, FreeBSD, Android, HP-UX, Solaris, ...

Co to jest Linux?

Jądro systemu

- Komunikacja ze sprzętem
- Zarządzanie uruchomionymi programami (procesami)

Powłoka/narzędzia

- Wszystko co pozwala komunikować się z jądrem, np. uruchamiać programy
- Rodzaje: CLI, GUI
- Zwykle Bash, Zsh
- GUI: Gnome, KDE, XFCE...

Dystrybucja

- Całościowy system operacyjny składający się z jądra, powłoki i zestawu programów
- Np. Ubuntu, Debian, Fedora, CentOS, Arch Linux...

Organizacja plików w Linuksie

System plików Linuksa różni się od tego z systemu Windows

- Do rozdzielania katalogów używamy / (jak w internecie) a nie \
- Nie ma podziału na dyski typu C:, D:, ale wszystkie ścieżki zaczynają się od głównego katalogu /
- Dyski „montuje się” do katalogów, np. /mnt/hdd1
- Większość rzeczy w Linuksie „jest plikiem”

Ważne katalogi w Linuksie

Spotkasz je w większości dystrybucji

- /bin, /usr/bin, /usr/local/bin – pliki wykonywalne (pliki programów)
- /etc – pliki konfiguracji systemu
- /home/*user* – katalog domowy użytkownika
- /var/log – katalog z plikami logów (dzienników systemowych)
- /tmp – pliki tymczasowe

Użytkownicy w Linuksie

Użytkownik

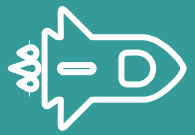
- Każda operacja w systemie musi być wykonywana jako jakiś użytkownik
- Każdy plik i katalog należy do jakiegoś użytkownika
- Użytkownik ma swój identyfikator (id)
- Administrator nazywa się root (id=0)

Grupa

- Każdy użytkownik może (ale nie musi) należeć do jednej lub wielu grup
- Każdy plik i katalog należy do jakiejś grupy
- Grupa ma swój identyfikator (id)

CLI

Command Line Interface



Konsola to podstawowy sposób działania z systemem Linux
Za pomocą poleceń konsoli można też pisać skrypty, czyli programy wykonywane w linii poleceń

Schemat polecenia konsoli

nazwa_polecenia

Nazwa polecenia może być też ścieżką do programu – wiele poleceń jest w zasadzie plikiem wykonywalnym.

Np. `/bin/ls`

-o wartość_opcji

W drugiej kolejności podajemy opcje polecenia. Opcje często występują w dwóch postaciach:

długiej, np.: --opcja

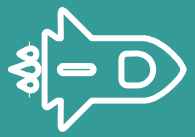
i krótkiej, np.: -o

parametr

Do części poleceń można podać na końcu dodatkowe argumenty, np.. nazwę pliku lub katalogu na którym ma operować polecenie

Przykład polecenia: `ls -al /var/log`

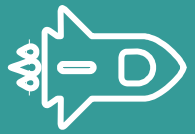
Linia zachęty



```
root@b448300499ae:/home#
maciej@maciej-ThinkPad-T430:/var$
```

Zwykle zawiera: nazwę użytkownika, nazwę komputera, ścieżkę i znak zachęty (zwyczajowo `$` dla użytkownika, `#` dla roota)

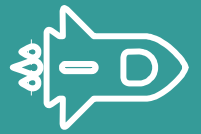
Tabulacja!



Bash automatycznie podpowiada nazwy poleceń i ścieżki po naciśnięciu tabulacji. Przy odpowiedniej konfiguracji może też podpowiadać parametry poleceń

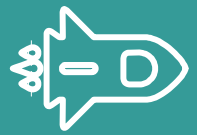
Używaj tego, by uniknąć literówek i przyspieszyć swoją pracę!

pwd



Informacja o bieżącym katalogu

cd



Zmiana katalogu

cd /home ← zmiana do katalogu /home

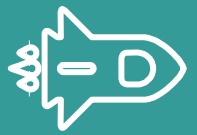
cd ~ ← zmiana do katalogu domowego

cd .. ← zmiana do katalogu nadrzędnego

cd . ← zmiana do katalogu bieżącego (brak zmiany)

cd ← zmiana do katalogu domowego

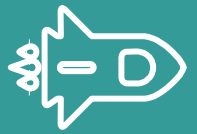
ls



Listowanie katalogu

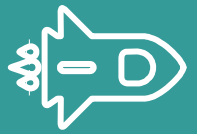
ls ← proste listowanie bieżącego katalogu
ls /var/log ← listowanie katalogu
ls -a ← listowanie wraz z plikami ukrytymi
ls -l ← listowanie wraz z informacją o dacie i uprawnieniach
ls -1 ← widok jednej kolumny

cat



Wyświetlenie zawartości pliku

touch

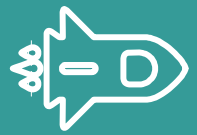


Założenie pustego pliku
Zmiana dat plików

Alternatywa: uśmiechnąć się do pliku:

```
:> nazwa_pliku
```

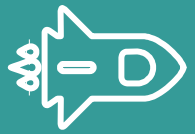
rm



Usunięcie pliku

rm -r ← usunięcie katalogu
rm -rf ← usunięcie katalogu (wymuszone)

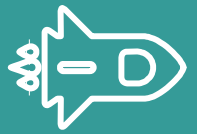
mkdir



Utworzenie katalogu

rmdir ← usunięcie **pustego** katalogu

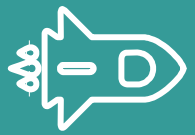
grep



Filtrowanie wejścia lub plików
Wyszukiwanie w plikach

```
grep Unpack /var/log/bootstrap.log
grep -R Unpack /var/log
```

man



Pokazuje podręcznik użytkownika danego polecenia

```
man ls
man grep
ls -h
grep --help
```

Przydatne polecenia

- less – przeglądanie zawartości pliku
- tail / head – pokazanie końca / początku pliku
- cut – wyświetlenie części linii
- sort – sortowanie plików
- uniq – usuwanie duplikatów z list
- mc – manager plików
- sudo – wykonywanie poleceń jako administrator

Rodzaje wejść i wyjść

Standardowe wejście (stdin)

Służy do pobierania danych wejściowych dla programu. Domyślnie jest to klawiatura w konsoli

Standardowe wyjście (stdout)

Służy do wyświetlenia wyniku działania programu. Domyślnie jest to ekran konsoli

Standardowe wyjście błędów (stderr)

Służy do wyświetlania informacji o błędach w działaniu programu, lub informacji diagnostycznych. Domyślnie jest to też ekran konsoli

Przekazywanie wyników poleceń

Każde z wejść i wyjść może zostać zmienione za pomocą znaków < i >

- **echo "foo" > ./plik** ← **przekierowuje wyjście stdout do pliku**
- **cat < ./plik** ← **używa pliku ~/plik jako stdin**

- ls 2> ~/plik ← przekierowuje stderr do pliku
- ls 2>&1 ← przekierowuje stderr do stdout
- ls > ~/plik 2>&1 ← stderr do stdout i całe wyjście do pliku

Łączenie wyników poleceń

Polecenia możemy ze sobą łączyć za pomocą znaku | (pipeline)

```
ls /var/log | grep last
ls -1 | wc -l
```

Uruchamiamy Dockera!

Możesz myśleć o Dockerze jak o wirtualnej maszynie – bardzo lekkiej i zwykle poświęconej pojedynczej usłudze

```
sudo docker run -it -d -p 2222:22 -p 8080:80
  ubuntu

sudo docker ps

sudo docker exec -it nazwa_kontenera bash

sudo docker kill nazwa_kontenera

sudo docker rm nazwa kontenera
```

Ćwiczenie 1

1. Sprawdź w jakim katalogu jesteś
2. Przejdź do katalogu `/bin`
3. Wylistuj go wraz z informacjami rozszerzonymi
4. Znajdź w nim wszystkie pliki zawierające frazę „`bash`”
5. Załóż katalog `/tmp/test`
6. Zapisz do pliku `/tmp/test/lista` listę plików bieżącego katalogu
7. Wyświetl plik `/tmp/test/lista`
8. Usuń plik `/tmp/test/lista`
9. Usuń katalog `/tmp/test`

Edycja plików - Nano

- Prosty edytor tekstowy
- Obsługa za pomocą skrótów klawiszowych Ctrl+… oraz Alt+...
- Pomoc wyświetla się na dole ekranu (^ oznacza Ctrl, M- oznacza Alt)
- Uruchamiamy go poleceniem `nano /sciezka/do/pliku`

Ćwiczenie 2

1. Zapisz do pliku /tmp/lista listę plików bieżącego katalogu
2. Otwórz ten plik w edytorze nano
3. Zmień coś, zapisz zmiany i wyjdź
4. Sprawdź poleceniem cat czy zmiany zostały zapisane

Uprawnienia w Linuksie - przydział

Użytkownik

Zbiór uprawnień dotyczący właściciela pliku/katalogu

Grupa

Zbiór uprawnień dotyczący grupy właścicielskiej

Inni

Zbiór uprawnień dotyczący użytkowników którzy nie są właścicielami i nie należą do grupy właścicielskiej

Uprawnienia w Linuksie - rodzaj

Odczyt (4 lub r)

Nadanie tego uprawnienia umożliwia odczyt zawartości pliku lub listowanie katalogu

Zapis (2 lub w)

Nadanie tego uprawnienia umożliwia zmianę zawartości pliku lub tworzenie plików w katalogu

Wykonanie (1 lub x)

Nadanie tego uprawnienia umożliwia wykonywanie pliku jako programu lub odczyt dodatkowych informacji o plikach w katalogu

Arytmetyka uprawnień

Uprawnienia składają się z trzech liczb dla każdego rodzaju dostępu:
użytkownik, grupa, inni

Wynikowe uprawnienie zapisuje się w postaci jednej liczby trzycyfrowej,
np. 755

Ćwiczenie 3

1. Załóż katalog `/tmp/testmod`
2. Zapisz do pliku `/tmp/testmod/lista` listę plików bieżącego katalogu
3. Ustaw uprawnienia do pliku `/tmp/testmod/lista` tak, by tylko właściciel mógł go odczytać i zapisać i nikt nie mógł go wykonywać (polecenie chmod *uprawnienie plik*)

```
-rw------- 1 root root 85 Nov 14 21:16 /tmp/testmod/lista
```

4. Ustaw uprawnienia do katalogu `/tmp/testmod` tak, by właściciel miał pełne uprawnienia, grupa mogła go odczytywać i listować, nastomiast pozostali nie mieli do niego żadnych uprawnień

```
drwxr-x--- 2 root root 4096 Nov 14 21:16 testmod
```

Instalacja programów

- Większość dystrybucji Linuksa posiada jakiś system pakietów
- W Ubuntu programy instalujemy poleceniem „apt” lub „apt-get”
  - „apt update” uaktualnia listę dostępnych pakietów
  - „apt install nazwa-pakietu” pobiera i instaluje pakiet
  - „apt search pakiet” wyszukuje pakiety

Ćwiczenie 4

1. Uaktualnij listę dostępnych pakietów
2. Zainstaluj pakiety:
   - nano
   - vim
   - man
   - mc
3. Uruchom program „mc”
4. Zobacz co widzisz, przejdź strzałkami po plikach
5. Wyjdź z niego wpisując „exit” i zatwierdzając enterem

Edycja plików - Vim

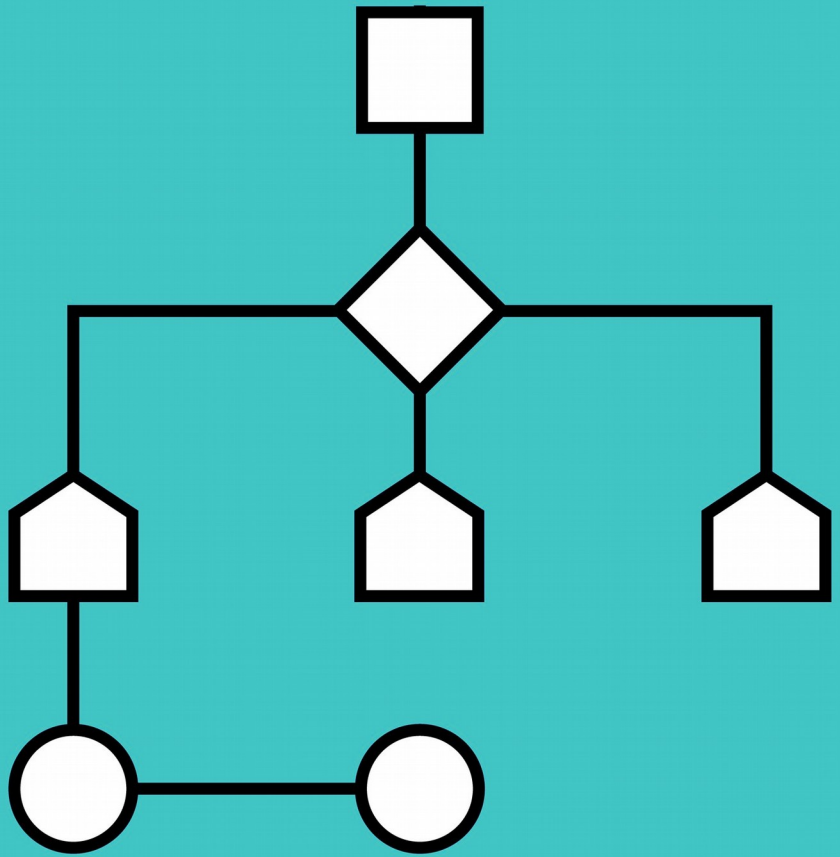
- Bardzo rozbudowany edytor tekstowy
- Szeroko rozpowszechniony – jego odmiany znajdziesz na wielu serwerach
- Kilka trybów – wpisywania, zamiany, zaznaczania (wizualny)
  - ESC – wyjście z trybu
  - I, R, V – wejście w tryby (wpisywania, zamiany, wizualny)
- :q – wyjście
- :q! - wyjście bez zapisu
- :w – zapis zmian
- :x lub :wq – wyjście z zapisem

Ćwiczenie 5

1. Zapisz do pliku /tmp/lista listę plików bieżącego katalogu
2. Otwórz ten plik w edytorze vim
3. Zmień coś, zapisz zmiany i wyjdź
4. Sprawdź poleceniem cat czy zmiany zostały zapisane

Bash jako język programowania

Konsola Linuksa udostępnia wiele małych narzędzi. Zapisując ich wywołania w pliku tworzymy skrypty, czyli de facto programy



Struktura skryptu

```
#!/bin/bash

#komentarz

echo "Hello world!"
```

Skrypt powinien być wykonywalny:

```
chmod +x skrypt.sh
```

Kolejność wykonywania poleceń

`polecenie1 && polecenie2` – wykonuje polecenie2 jeśli polecenie1 zakończyło się sukcesem

`polecenie1 || polecenie2` – wykonuje polecenie2 jeśli polecenie1 zakończyło się niepowodzeniem

`polecenie1 ; polecenie2` – wykonuje polecenie1 a następnie polecenie2

Zmienne

```
#!/bin/bash

ZMIENNA="wartość"

ZMIENNA=`echo "foo"`

ZMIENNA=$(echo "foo")

ARR=(element1 element2)

echo $ZMIENNA

echo ${ARR[0]}
```

Zmienne specjalne

$0 – nazwa skryptu

$1...9 – nazwa kolejnego parametru wywołania

$@ - wszystkie parametry

$? - kod wyjścia ostatniego polecenia

Ćwiczenie 6

1. Napisz skrypt /tmp/skrypt.sh który przyjmie jeden parametr i wyświetli go na ekranie
2. Uruchom swój skrypt
3. Sprawdź jaki był kod wyjścia

Instrukcje warunkowe „if”

```
if [ warunek ]; then
  echo "warunek spełniony”
else
  echo "warunek nie jest spełniony"
fi
```

Jak pisać warunki? Zobacz: `man test`

Pętla „for” i „while”

```
for ZMIENNA in 1 2 3 4; do
  echo $ZMIENNA
done
while [ $ZMIENNA -lt 10 ]; do
  ZMIENNA=$((ZMIENNA+1))
  echo $ZMIENNA
done
```

Ćwiczenie 7

1. Napisz skrypt /tmp/skrypt2.sh który:
2. Jeśli pierwszy parametr jest równy "ala”, wyświetli

   "Ala ma kota"
3. Jeśli pierwszy parametr jest równy "kot”, wyświetli

   "miau"
4. W przeciwnym wypadku powinien dla każdego podanego parametru wypisać tekst:

   Ala ma *treść parametru*

Procesy w Linuksie

- Proces = uruchomiony program
- Identyfikator procesu – liczba PID
- Procesy układają się w drzewa – każdy ma „rodzica” i może mieć „dzieci”
  - Rodzicem wszystkich procesów jest `init` *(PID=1)*
- Procesy mogą działać „pierwszoplanowo” (*foreground*) albo „w tle” (*background*)
  - Proces w tle możemy uruchomić przez zakończenie linii poleceń znakiem „`&`”
- Procesami sterują sygnały (*signal*) wysyłane najczęściej poleceniem „kill”

Listowanie procesów

- Polecenie `ps` pokazuje listę programów
- Posiada wiele opcji, ale warto zapamiętać: „`ps aux`” pokazuje wszystkie procesy wszystkich użytkowników

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 225776  4884 ?        Ss   lis19   0:15 /sbin/init splash
```

- Polecenie „top” pokazuje procesy uporządkowane według używanych zasobów

Zarządzanie procesami - sygnały

- Podstawowe rodzaje sygnałów:
  - SIGTERM (15) – kończenie procesu z powiadomieniem go o tym fakcie (proces może zignorować ten sygnał)
  - SIGKILL (9) – kończenie procesu bez powiadamiania go
  - SIGINT (2) – przerwanie procesu, najczęściej wysyłane przez Ctrl+C w konsoli
  - SIGTSTP (20) – zatrzymanie procesu (wysyłane przez Ctrl+Z)
    - Proces można wznowić poleceniem `fg` (foreground) lub `bg` (background), które wysyłają sygnały

Zarządzanie procesami - przykłady

- Znalezienie procesu:
  - `ps aux | grep nazwa_procesu`
- Zabicie procesu:
  - `kill id_procesu`
  - `kill -9 id_procesu`
- Zabicie wszystkich procesów danego programu:
  - `killall nazwa_programu`

Ćwiczenie 8

1. Sprawdź jakie procesy są uruchomione poleceniem „`ps aux`”
2. Wykonaj polecenie „`sleep 3600`”
3. Zatrzymaj wykonanie klawiszami Ctrl+C i sprawdź znowu „`ps aux`”
4. Wykonaj polecenie „`sleep 3600 &`”
5. Zatrzymaj wykonanie Ctrl+Z i sprawdź „`ps aux`”
6. Zabij działający proces „sleep 3600” poleceniem „`kill nr_procesu`”

Zarządzanie procesami - lsof

- Polecenie „lsof” odpowiada nam na pytanie „kto trzyma otwarty plik”
- Pozwala też sprawdzić kto otworzył port
- Przykłady:
  - `lsof sciezka_pliku` – znalezienie programu który otworzył plik
  - `lsof -i:8080` – znalezienie programu który otworzył port 8080

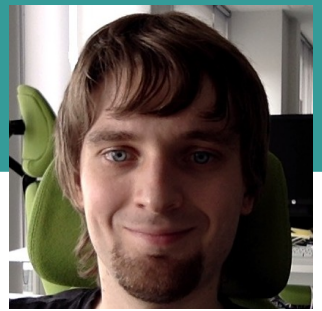
Harmonogram zadań - cron

- Cron pozwala wykonywać zadania cyklicznie, np. co minutę, co godzinę, w każdy poniedziałek, w każdy 3 i 15 dzień miesiąca, itp.
- Jego konfiguracja to plik tekstowy z wpisami typu:

```
30      02      *       *       *           /usr/local/bin/do-backup.sh
```

Harmonogram zadań - cron

Minuta (0 - 59)	Godzina (0 - 23)	Dzień miesiąca (1 - 31)	Miesiąc (1 - 12)	Dzień tygodnia (0 - 6)	Polecenie
*	*	*	*	*	ls
0	1	*	*	*	tar
0	0	10	1,2,3	*	wget
*/5	*	*	*	1-5	gzip



Dzięki

You can find me at
@mkosiedowski & mkosied@gmail.com