Potokowanie strumienia danych.

Wyjście jednego polecenia może być użyte jako wejście do innego przez użycie potoku (pipe, symbol „|”).

składnia:command1|command2

W pojedynczym potoku (pipe) może istnieć maksymalnie 4 KB nieprzetworzonych danych. Jeżeli proces generujący wyjście próbuje zapisać dane do zapełnionego potoku, jest to zatrzymywane i wznawiane w chwili, gdy zapis jest już możliwy. Z drugiej strony – proces czytający jest zatrzymywany, gdy próbuje czytać z pustego potoku.

Ćwiczenie 1

1.
```
root@bolek-VirtualBox:~# ls -l /etc |less
drwxr-xr-x  5 root root    4096 kwi 23 12:59 xdg
-rw-r--r--  1 root root     477 gru 20 19:20 zsh_command_not_found
(END)
```

Czasami możemy chcieć wynik polecenia mieć zarówno wyświetlony na ekranie jak i zapisany do pliku. Użyjemy wtedy polecenia tee:

2.
```
bolek@bolek-VirtualBox:~$ ls -l |tee out
razem 56
drwxr-xr-x 5 bolek bolek 4096 maj 19 12:02 baza
```

W tym przypadku wyjście polecenia zostanie wyświetlone na ekranie i jednocześnie zapisane do pliku output.

Aby przekierować wyjście sekwencji kilku poleceń w wierszu poleceń, polecenia te muszą być oddzielone od siebie średnikami i cała sekwencja zamknięta nawiasami (command1; command2):

3.
```
bolek@bolek-VirtualBox:~$ (id;ls ~) > out
bolek@bolek-VirtualBox:~$ cat out
uid=1000(bolek) gid=1000(bolek) grupy=1000(bolek),4(adm),24(cdrom),27(sudo),30(d
ip),46(plugdev),118(lpadmin),128(sambashare)
baza
bbhold
```

Zgłoszenie 1

Ćwiczenie 2

Polecenie wc (word count) zlicza słowa w pliku tekstowym. Polecenie wc –l zlicza wiersze w pliku.

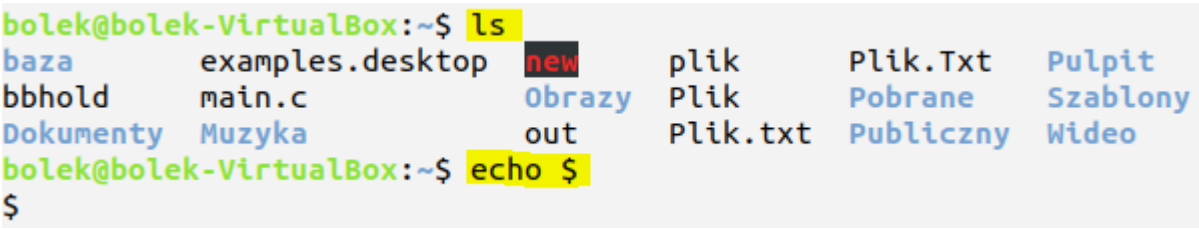
Przekaż potokiem wyjście polecenia ls -l do polecenia wc -l. Co będzie rezultatem takiej konstrukcji?

1. 

Powłoka uruchamia osobną podpowłokę (subshell) dla przetwarzania indywidualnych poleceń.
By przekierować powiązane polecenia, powłoka musi być zmuszona do wykonywania łańcucha powiązanych poleceń w tej same podpowłoce. Jest to robione przez zamknięcie wyrażenia w nawiasach.

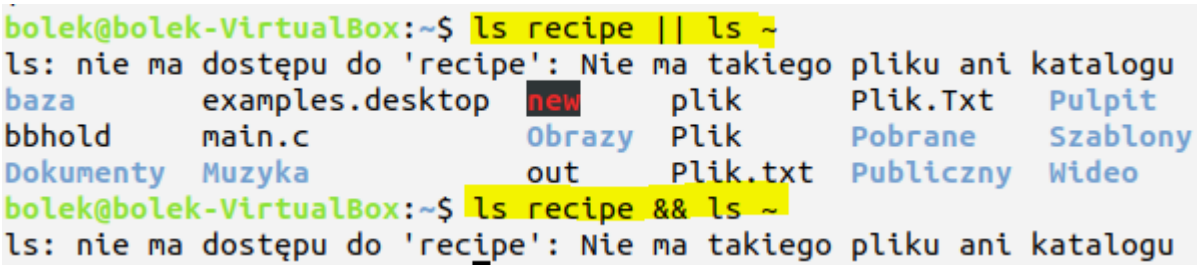
Po wykonaniu, każdy program zwraca wartość stanu wykonania (status). Jeżeli ta wartość jest równa zeru – program zakończył się sukcesem. W przypadku błędu – zwracana jest wartość różna od zera.
Zależnie od programu – różne wartości wskazują na różne błędy (są kodami błędu).

Można użyć polecenia echo $? do wyświetlenia zwracanej wartości:

2. 

Zwracany kod może być użyty jako wyzwalacz (trigger) wykonania innego polecenia:

Wiązanie	***Wynik***
Polecenie1 && polecenie2	Polecenie 2 zostanie wykonane tylko wtedy, gdy polecenie1 zakończy się bezbłędnie.
Polecenie1 || polecenie2	Polecenie 2 zostanie wykonane tylko wtedy, gdy polecenie1 zakończy się z błędem.

3. 

Wyjaśnienie. Plik recipe nie istnieje i polecenie ls recipe musi zakończyć się błędem.

false && echo tak

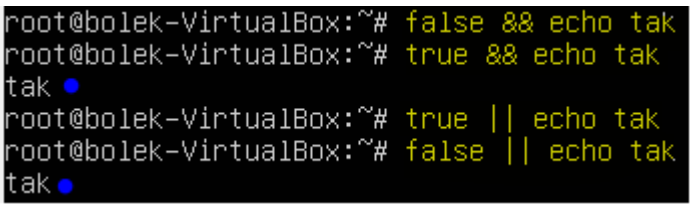
true && echo tak

tak

true || echo tak

false || echo tak

tak

4. 

Zgłoszenie 2

Ćwiczenie 3

Stosowanie potokowania i przekierowywania I

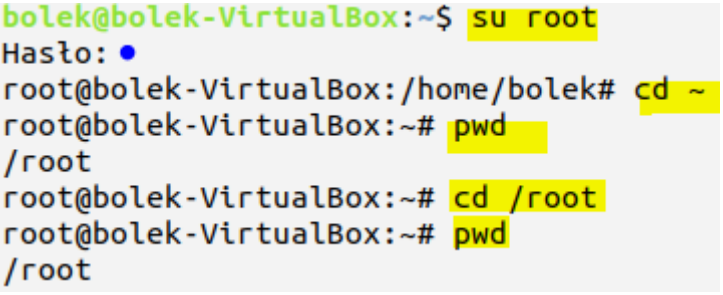
Jaka wartość jest zwracana po wykonaniu poniższych poleceń?

Polecenie wc zlicza słowa w pliku tekstowym. Polecenie wc –l zlicza wiersze w pliku.

ls – l | wc – l

1. 

cd /root

2. 

Odczytaj plik o nazwie /var/log/syslog.1

less /var/log/syslog.1

3. 

Wyświetlenie zawartości katalogu z określeniem typu i kodowania plików.

4.
```
bolek@bolek-VirtualBox:~$ su bolek
Hasło:
bolek@bolek-VirtualBox:~$ file * |less
```

```
baza:              directory
bbhold:            regular file, no read permission
Dokumenty:         directory
examples.desktop:  UTF-8 Unicode text
main.c:            C source, ASCII text
```

q

echo $LOREM

5.
```
root@bolek-VirtualBox:/home/bolek# echo $LOREM
```

echo $?

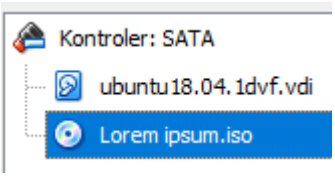
6.
```
root@bolek-VirtualBox:/home/bolek# echo $?
0
```

Wyrażenia regularne

Wyrażenia regularne to ciągi znaków i metaznaków, wzorce opisujące łańcuchy symboli.

Podłącz plik Przykłady użycia poleceń grep oraz egrep:

7.
```
root@bolek-VirtualBox:~# touch plik1
root@bolek-VirtualBox:~#
root@bolek-VirtualBox:~# touch plik2
root@bolek-VirtualBox:~# ls plik*
plik1  plik2
root@bolek-VirtualBox:~# echo blurb > plik1
root@bolek-VirtualBox:~# echo Blurb > plik2
root@bolek-VirtualBox:~# egrep '(b|B)lurb' plik*
plik1:blurb
plik2:Blurb
root@bolek-VirtualBox:~# grep '(b|B)lurb' plik*
root@bolek-VirtualBox:~#
```

8. Lorem ipsum.iso 

Zgłoszenie 3

Ćwiczenie 4

```
root@bolek-VirtualBox:/home/bolek# grep "Lipsum" "/media/bolek/20160407_175510/L
orem ipsum.txt"
Lipsum
Lipsum
Lipsum
Lipsum
```

Sprawdź i podaj swój numer urządzenia (niebieska kropka)

Co znaczą podane niżej wyrażenia regularne? Jak dużo wierszy je spełnia z pliku loremipsum.txt? (pobierz loremipsum.txt)

^.t

1.
```
root@bolek-VirtualBox:/home/bolek# grep ^.t "/media/bolek/20160407_175510/Lorem
ipsum.txt"
Etiam consequat dolor tempor arcu rhoncus pharetra. Vestibulum ante ipsum primis
 in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean tellus eros,
```

it\>

2.
```
root@bolek-VirtualBox:/home/bolek# grep it\> "/media/bolek/20160407_175510/Lorem
 ipsum.txt"
```

m{2}

3.
```
root@bolek-VirtualBox:/home/bolek# grep m{2} "/media/bolek/20160407_175510/Lorem
 ipsum.txt"
```

[^.,]$

4.
```
root@bolek-VirtualBox:/home/bolek# grep [^.,]$ "/media/bolek/20160407_175510/Lor
em ipsum.txt"
```

Zgłoszenie 4

Ćwiczenie 5

Utwórz wyrażenia regularne opisujące podane kryteria. Użyj przykładowego pliku loremipsum.txt.

1. Wszystkie wiersze z przynajmniej jedną dużą literą.
2. Wszystkie wiersze zawierające słowo „in” lub „ad”.
3. Wszystkie wiersze, w których występuje przynajmniej jedno słowo trzyliterowe.

Zgłoszenie 5

Ćwiczenie 6

ls -l – zwraca zawartość katalogu

1.
```
root@bolek-VirtualBox:~# ls -la
razem 40
drwx------  6 root root 4096 maj 19 14:10 .
drwxr-xr-x 24 root root 4096 kwi 23 12:58 ..
```

ls -l | awk ‘{print $1 $8}’ – filtruje 1 i 8 kolumnę

2.
```
root@bolek-VirtualBox:~# ls -l | awk '{print $1 $8}'
razem
-rw-r--r--14:12
```

ls -l | awk ‘{print $1 ” ” $8}’ – filtruje 1 i 8 kolumnę rozdzielając je spacją

3.
```
root@bolek-VirtualBox:~# ls -l | awk '{print $1 " " $8}'
razem
-rw-r--r-- 14:12
```

Zgłoszenie 6