

SO Temat: 4 . Podsystem plików – przeszukiwanie i kontrola zasobów.

Na zajęciach: Podsystem plików – przeszukiwanie i kontrola zasobów, jak system wyszukuje zasób na poziomie fizycznym w oparciu o ścieżkę dostępu.

Ćwiczenia: zarządzanie zasobami użytkownika w ramach SO Linux. Strumienie WE, WY, WY błędów. Łączenie poleceń (potoki poleceń). Przeszukiwanie zasobów - polecenie find. Systemy pomocy w Linuxie – man, info.

Zawartość

| | |
|---|---|
| Mechanizm WE/WY - strumienie..... | 1 |
| łączenie poleceń (pipeline – strumień poleceń)..... | 3 |
| Polecenie find..... | 3 |
| Polecenie locate..... | 5 |
| ZADANIA:..... | 5 |

Jednym z przydatnych poleceń jest

`wc <==` licznik - przelicza dołączony tekst i podaje w wyniku – ile w nim jest wierszy, wyrazów i znaków, np.

`wc dane ==> 11 33 234`

Można wydzielać – tylko wiersze: `wc -l`, tylko wyrazy: `wc -w`, tylko znaki: `wc -c`.

Inne przydatne polecenie:

`echo komunikat <==` wyświetlanie komunikatu (domyślnie na ekranie).

Mechanizm WE/WY - strumienie.

Mamy urządzenia końcowe (monitor, konsola, klawiatura, mysz) oraz mechanizmy przekazujące na te urządzenia (strumienie).

Strumienie:

-WE - stdin - in

-WY - stdout -out

-WY błędów - stderr - err.

O jakie błędy chodzi i co trafia na to wyjście?

Błędy - (w pracy z konsolą) złe polecenia. Na WY błędów - będzie trafiała reakcja SO - shella => będą trafiały komunikat o błędach.

W danej chwili strumienie są ukierunkowane na konkretne urządzenia. Można to ukierunkowanie zmienić (przekierować strumień).

Strumienie - na jakie urządzenia:

-WE - klawiatura

-WY – konsola (ekran, monitor)

-WY błędów - konsola (ekran, monitor).

Przekierowanie:

-WY - z konsoli na plik "dane123":

`ls > dane123` <== znak ">" przekierowuje.

`echo tekst > plik22`

-WE - z klawiatury na plik "dane123":

`cat < dane123`

-WY błędów - z konsoli na plik "dane123":

`ls-l 2> dane123`

Info:

strumienie są traktowane jako pliki specjalne => mają przyporządkowane i-węzły - ich numery:

-WE – 0

-WY - 1

-WY błędów - 2.

Jak w jednym poleceniu przekierować:

-WY do jednego pliku, WY błędów do drugiego?

ls-l >gg 2>bledy <= WY normalne do pliku gg, WY błędów do pliku bledy.

-WY i WY błędów do jednego pliku?

ls-l >gg 2>&1

Uwaga:

-przekierowanie ">" powoduje stworzenie nowej zawartości pliku.

--przekierowanie ">>" powoduje dopisanie (na końcu) do istniejącej treści.

Za pomocą poniższej sekwencji działań można stworzyć plik tekstowy "dane":

cat > dane

<treść>

CTRL+Z (polecenie zapisu i zamknięcia = program cat pozostaje uruchomiony).

Polecenie:

echo <komentarz>

powoduje wyświetlenie komentarza na monitorze.

Można przekierować ten komentarz do pliku, np.:

echo CZESC > dane

Np.:

echo ls -l

echo 'ls -l', echo „ls -l”

echo `ls -l` <= “krzywe apostrofy – pod tyldą” – wykonanie polecenia

echo „\$HOME” <= zawartość zmiennej

echo \$HOME <= zawartość zmiennej

echo '\$HOME’ <= tekst

Np.:

echo "Zawartość katalogu domowego:" `ls` "a nazwa katalogu domowego=\$HOME"

Można też za pomocą strumieni WY połączyć kilka plików i zapisać go do jednego – jeżeli są pliki dane1, testy22 i list11 to polecenie:

cat dane1 testy22 list11 > archiwum

spowoduje ich zapis do pliku archiwum.

Polecenia: *head* i *tail* pokazują 10 pierwszych i 10 ostatnich linii pliku (można ilość wyświetlanych linii zmienić), np.:

head dane1

Ale można też wykorzystać przekierowanie strumienia WE:

head < dane1

head < dane1 testy22 list11 > razem .

Polecenie:

cat * 1> dane 2>bledy

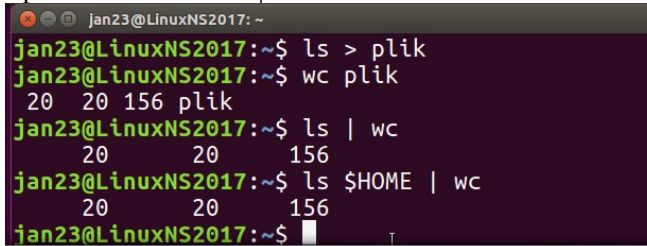
Łączenie poleceń (pipeline – strumień poleceń).

Wynik działania poprzedzającego polecenia staje się danymi WE dla następnego. Składnia:

<polecenie> | <polecenie> | <polecenie> ...

Np.: chcemy policzyć ile plików i katalogów jest w domowym katalogu – wyświetlanie zawartości katalogu – polecenie `ls`, przeliczanie – ile wierszy, słów i znaków jest w pliku – polecenie `wc`.

`ls > plik` WY WE
`wc plik` =====> `ls | wc -w`



```
jan23@LinuxNS2017: ~  
jan23@LinuxNS2017:~$ ls > plik  
jan23@LinuxNS2017:~$ wc plik  
20 20 156 plik  
jan23@LinuxNS2017:~$ ls | wc  
20 20 156  
jan23@LinuxNS2017:~$ ls $HOME | wc  
20 20 156  
jan23@LinuxNS2017:~$
```

Polecenie `wc` (word counter) - liczy wiersze, słowa, znaki, np.:

`wc dane123`

(`wc -l`, `wc -w`, `wc -c`, np.:

`wc -l dane123`).

Łączenie poleceń - symbol "|":

`ls -l | wc -l`

Wygodnym poleceniem jest polecenie:

`less` <= stronicuje zawartość pliku.

Np.: polecenie

`ps -A` <=pokazuje uruchomione zadania w SO

`ps -A | less` <= pokazuje ze stronicowaniem (przeglądanie – strzałki góra-dół), WY – naciskamy 'q'.

Polecenie find.

Polecenie `find` - służy do wyszukiwania zasobów (różnych plików) w-g różnych kryteriów.

Składnia:

`find <gdzie> <kryteria>`

np.:

`find . -name dane123`

`find . -iname "a*"`

Kryteria:

`-amin`, **`-atime`**, **`-cmin`**, **`-ctime`**, **`-mmin`**, **`-mtime`** - szukanie zasobów: `-a` – do których był dostęp, `-c` - które były zmieniane, `-m` - zawartość których była modyfikowana,

podaje się też przedział czasowy: `min` - minuty, `time` - dni,

podaje się też ile jednostek np. `-5`, `5`, `+5` <= mniej niż 5, dokładnie 5, więcej niż 5

np.:

`zada`

Przykład – o 14.30 został utworzony plik `plik1` , plik `KKKK` został utworzony o 15.00.

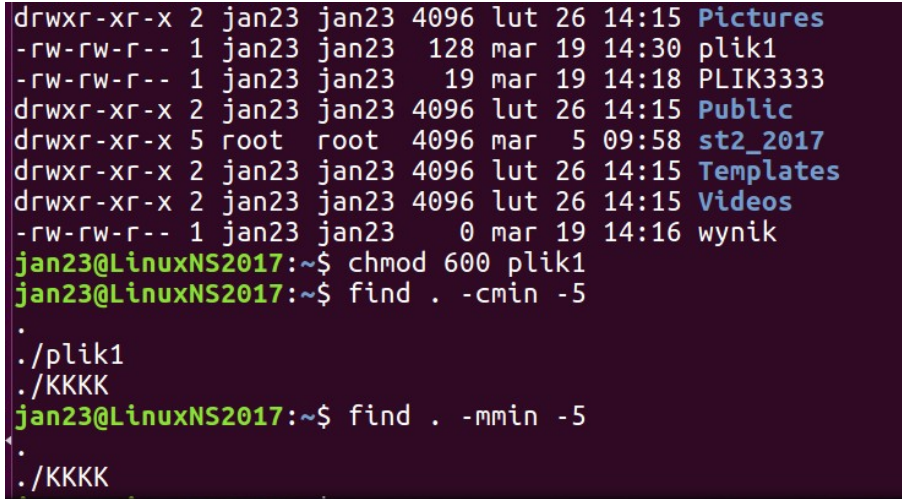
O 15.02 dokonano zmiany praw dostępu do pliku *plik1*, o 15.03 wydano polecenie:

```
find . -cmin -5
```

o 15.04 polecenie:

```
find . -mmin -5
```

Wyniki – zrzut ekranu:



```
drwxr-xr-x 2 jan23 jan23 4096 lut 26 14:15 Pictures
-rw-rw-r-- 1 jan23 jan23 128 mar 19 14:30 plik1
-rw-rw-r-- 1 jan23 jan23 19 mar 19 14:18 PLIK3333
drwxr-xr-x 2 jan23 jan23 4096 lut 26 14:15 Public
drwxr-xr-x 5 root root 4096 mar 5 09:58 st2_2017
drwxr-xr-x 2 jan23 jan23 4096 lut 26 14:15 Templates
drwxr-xr-x 2 jan23 jan23 4096 lut 26 14:15 Videos
-rw-rw-r-- 1 jan23 jan23 0 mar 19 14:16 wynik
jan23@LinuxNS2017:~$ chmod 600 plik1
jan23@LinuxNS2017:~$ find . -cmin -5
.
./plik1
./KKKK
jan23@LinuxNS2017:~$ find . -mmin -5
.
./KKKK
```

-empty - puste pliki

find . -empty

```
mirek@orfi:~$ touch PUSTY
```

```
mirek@orfi:~$ ls -l Y
```

```
-rw-r--r-- 1 mirek mirek 0 03-24 14:35 PUSTY
```

```
mirek@orfi:~$ > PUSTY2
```

```
mirek@orfi:~$ ls -l PUSTY2
```

```
-rw-r--r-- 1 mirek mirek 0 03-24 14:36 PUSTY2
```

Pusty plik - nie ma zawartości - jego rozmiar = 0.

-maxdepth <ile>, -mindepth <ile> <= do jakiego (od jakiego) poziomu zagłębienia w podkatalogi przeszukujemy (ten parametr piszemy jako pierwszy w poleceniu)

np.: find . -maxdepth 1 -empty

-maxdepth 3

-name <nazwa> <= w-g nazwy

-iname <nazwa> <= w-g nazwy (można z ?,*)

-inum <numer> <= pliku z podanym nr-em i-węzła

-perm <prawa dostępu>, np. -perm 741

np.:

find . -perm -u=x <= pliki dla których właściciel ma prawo x – to nie wyklucza innych praw; (u - user, g - grupa, o - reszta - jak w chmod)

-type <typ>, np.: -type f - zwykłe pliki

(typy: f - zwykły, d - katalog, l - dowiązanie symboliczne, b - urz. blokowe, c - urz. znakowe, s - socket (gniazdo), p - pipe, ...)

urz. blokowe - wymiana danych z nim odbywa się blokami,

urz. znakowe - wymiana danych z nim odbywa się znakami,

np.:

find /dev -type b
sda <= 1-szy dysk SCSI
sdb, sdc
hda <= 1-szy dysk IDE
sda1 <= 1-sza partycja na 1-szym dysku
sdc1, sdc2, itd
-user <identyfikator> <= plików podanego użytkownika (podaje się login lub jego nr id).

Wynikiem polecenia find jest tekst.

Żeby dokonać operacji na wyszukanych plikach trzeba użyć:

-exec <polecenie> {} \; (lub **-ok <polecenie> {} \;**).

np.:

find . -maxdepth 1 -type f -exec cat {} \;

(**find . -empty -exec ls -l {} \;**)

-ok <polecenie> {} \; <= shell będzie wymagał potwierdzenia wykonania polecenia dla każdego zasobu oddzielnie, np.:

find . -maxdepth 1 -type f -exec rm {} \;

Polecenie locate.

Podobne do find polecenie:

locate <parametry> <wzorzec>

Polecenie nie szuka samych plików, przeszukuje indeks plików tworzony cyklicznie przez system.

Przykład:

locate -b 'aa' (wcześniej plik został utworzony w bieżącym katalogu).

ZADANIA:

1. Wykorzystując mechanizm strumieni WE/WY:

a- do pliku dane22 zapisać informację o pogodzie – **bez edytora**, dopisać wynik polecenia *ls -l*, następnie dopisać nazwę bieżącego katalogu;

b- do pliku drzewo zapisać strukturę katalogu domowego, następnie pliki dane 22 i drzewo zapisać do pliku wynik;

c- wykorzystując mechanizm WE zapisać do jednego pliku razem zawartość plików dane22, drzewo, wynik;

d- policzyć ilu użytkowników pracuje teraz na serwerze;

e- policzyć ile plików i podkatalogów jest w danym katalogu.

2. Wykorzystując polecenie find znaleźć:

a- wszystkie puste pliki zwykłe w katalogu domowym i jego podkatalogach do poziomu 1;

find \$HOME -maxdepth 1 -empty -type f

b- znaleźć w katalogu /dev wszystkie pliki opisujące pseudourządzenia znakowe, policzyć – ile ich jest;

find /dev -type c

c- wyszukać wszystkie podkatalogi w katalogu domowym z prawami dostępu r-x dla właściciela;

find \$HOME -maxdepth 1 -type d -perm -u=rx <= NIE WYKLUCZA PRAWA W

d- wyszukać wszystkie pliki zwykłe katalogu bieżącym o nazwach zaczynających się na jedną z liter a-f, w i podać polecenie wyświetlenia ich zawartości;

find ./[a-f,w] -type f -exec cat {} \;*

e- wyszukać w podkatalogach do poziomu 3 w katalogu domowym pliki **niepuste** **utworzone** lub modyfikowane nie wcześniej niż 30 minut temu.

f- wyszukać wszystkie pliki zwykłe:

- do których był dostęp przez ostatnie 30 min.,

- które były modyfikowane przez ostatnie 30 min.,

- których zawartość była zmieniana przez ostatnie 30 min.

3. Wykorzystując łączenie poleceń:

a– policzyć ile urządzeń blokowych jest zdefiniowanych w katalogu /dev;

b- policzyć ile wierszy razem zawierają pliki o nazwach zaczynających się na jedną z liter a,c-e,w;

c- wyszukać nazwy wszystkich podkatalogów od poziomu 2 do 3 w katalogu głównym i policzyć ile ich jest.