

Edytor strumieniowy sed

# PRZETWARZANIE TEKSTU

Praca z edytorem tekstu może być żmudna, jeśli musisz wykonywać wielokrotnie te same zadania lub wywoływać takie same polecenia dla różnych plików.

Edytor sed oszczędzi Ci wiele pisania.

**HEIKE JURZIK**

**S**trumieniowy Edytor jest popularnym edytorem tekstowym w systemach Unix i Linux. W odróżnieniu od edytorów takich jak Vi czy Emacs, sed nie jest on interaktywny. Można w nim natomiast zdefiniować tekst do usunięcia, modyfikacji czy wstawienia z wiersza poleceń. Da się również połączyć wiele poleceń seda i uruchamiać je jako pojedyncze zadanie wsadowe. Jeśli chcesz zmodyfikować wiele znaków w jednym lub wielu plikach, sed znacznie ułatwi to zadanie.

Edytor sed wczytuje zazwyczaj tekst ze **standardowego wejścia** (*stdin*), ale może też czytać z jednego lub wielu plików. Zastosuje on podane polecenia do każdego wiersza, zapisując go do bufora. Następnie - jeżeli nie zmieniono ustawień domyślnych - zawartość bufora zostanie przesłana do **wyjścia standardowego** (*stdout*). Ogólna składnia polecenia jest następująca:

```
sed [opcje] >
'polecenie(a)' plik(i)
```

Aby zabronić powłóce interpretowania poleceń przeznaczonych dla seda, należy umieścić je w pojedynczych cudzysłowach.

Polecenia mogą dotyczyć pojedynczych wierszy, części wierszy, ciągów znaków albo - jeśli żadne z powyższych nie zostanie wybrane - całych plików. Definicję wybierającą wiersze do edycji nazywamy *adresem*. W Tabeli 1 prezentujemy najczęściej używane polecenia *seda* i przykłady adresów.

## Proste wyjście

Jak wspomniano wcześniej, polecenia programu sed stosowane są do tak zwanych ad-

resów (konkretne wiersze albo sekcje pliku). Można łączyć adresy z wyrażeniami regularnymi, tworząc bardzo precyzyjne definicje. Poniższe sekcje i przykłady prezentują możliwości adresowania dla różnych poleceń *seda*; opcji tych używać można ze wszystkimi poleceniami edytora.

Polecenie *p* służy do prostego wypisywania wyjścia. Aby na przykład wyświetlić drugi wiersz, wstaw po prostu przed poleceniem cyfrę 2:

**Tabela 1: Polecenia seda**

Polecenie	Opis	Przykład
<b>p</b>	Wypisuje wiersze na ekranie.	<code>sed -n '1,10p' file</code> - wyświetla tylko pierwsze dziesięć wierszy pliku <i>file</i> .
<b>d</b>	Usuwa określony obszar pliku.	<code>sed '3d' file</code> - usuwa trzeci wiersz w pliku <i>file</i> .
<b>s</b>	Zastępuje ciągi znaków.	<code>sed 's/KDE/Gnome/' file</code> - zamienia wystąpienia słowa „KDE” na „Gnome” w całym pliku.
<b>a</b>	Dodaje tekst do określonego wiersza (użyteczne w skryptach).	<code>1a  -EOL-</code> <i>To jest nowy wiersz.</i> - dołącza tekst do pierwszego wiersza.
<b>i</b>	Wstawia tekst do określonego wiersza (użyteczne w skryptach).	<code>1i  -EOL-</code> <i>To jest nowy wiersz.</i> - wstawia tekst przed pierwszy wiersz.
<b>c</b>	Zamienia wiersze lub część wierszy (użyteczne w skryptach).	<code>2c  -EOL-</code> <i>To jest inny wiersz.</i> - zamienia drugi wiersz przez następujący po „c” tekst.
<b>r</b>	Przetwarza treść pliku i dołącza go pod podanym adresem.	<code>sed '2r new.txt' file</code> - dodaje zawartość pliku <i>new.txt</i> do drugiego wiersza w pliku <i>file</i> .
<b>w</b>	Zapisuje określone wiersze lub sekcje do nowego pliku.	<code>sed '5,\$w new.txt' file</code> - zachowuje wiersze od piątego (włącznie) do ostatniego w pliku <i>new.txt</i> .

```
$ sed '2p' file
To jest pierwszy wiersz.
Drugi wiersz.
Drugi wiersz.
To jest wiersz 3.
...
```

Rezultat prawdopodobnie nie jest dokładnie tym, czego oczekiwaliśmy: zamiast drugiego wiersza, na ekranie wyświetlana została cała zawartość pliku, a drugi wiersz jest wyświetlany dwa razy. Żeby usunąć niepożądane wiersze, po prostu dodaj opcję `-n`:

```
@Li:$ sed -n '2p' file
Drugi wiersz.
```

## Usuwanie

Polecenie usuwania (`d`) w swojej najprostszej postaci oczekuje numeru wiersza, który ma zostać przetworzony. Na przykład

```
sed '1d' file
```

usuwa pierwszy wiersz z pliku `file`. Aby usunąć wiele wierszy za jednym zamachem, należy podać numery pierwszego i ostatniego wiersza do usunięcia, rozdzielone przecinkiem - przykładowo polecenie

```
sed '2,4d' file
```

usunie wiersze od drugiego do czwartego. Równie proste jest usuwanie co `n`-tego wiersza. Po wywołaniu polecenia

```
sed '1~3d' file
```

usunięty zostanie co trzeci wiersz, zaczynając od wiersza pierwszego. Jeśli chcesz usunąć zawartość pliku, począwszy od piątego wiersza aż do końca pliku, nie potrzebujesz liczyć liczby wierszy i podawać sedowi konkretnych liczb. Zamiast tego możesz użyć wyrażenia regularnego. Po wywołaniu polecenia

```
sed '5,$d' example.txt
```

skasowane zostaną wszystkie wiersze od wiersza piątego aż do końca pliku, określonego przez znak dolara.

Polecenie usuwania jest bardzo użyteczne, jeśli chcesz obejrzeć pliki konfiguracyjne z katalogu `/etc`, które zawierają wiele komentarzy. Program `sed` pozwala usunąć wszystkie wiersze zaczynające się od znaku „#”.

Rysunek 1: Pliki konfiguracyjne bez komentarzy - sed to potrafi.

```
sed '/^#.*d' /etc/inetd.conf
```

Adres dla polecenia `d` jest wyrażeniem regularnym, umieszczonym pomiędzy dwoma ukośnikami. Wyrażenie to określa dowolny wiersz zaczynający się od znaku „#” i zawierający poza tym dowolne znaki. Puste wiersze z pliku konfiguracyjnego nadal pojawiają się na wyjściu, możesz zatem użyć innego wyrażenia regularnego (`^[^#].*`), żeby wypisać (polecenie `p`) wszystkie wiersze, które nie zaczynają się od znaku „#” (Rysunek 1).

## Podstawienia

Polecenie `s` pozwala na zamienianie ciągów znaków. Po poleceniu następuje znak separatora, poszukiwane słowo kluczowe, następny znak separatora, ciąg, który chcesz wstawić, i ostatecznie zamykający znak separatora. Separatorem może być w zasadzie każdy znak, nie może on jednak pojawiać się w szukanym słowie kluczowym. Aby zamienić wszystkie wystąpienia słowa „rząd” przez słowo „wiersz”, można zastosować następujące polecenie:

```
$ sed 's/rząd/wiersz/' file
To jest pierwszy wiersz.
To jest drugi wiersz, a to 2
kończy ten rząd.
...
```

Edytor `sed` zamienił tylko pierwsze wystąpienie słowa kluczowego w każdym wierszu. Można jednak dodać przełącznik `g`, aby zamienić wszystkie wystąpienia:

```
sed 's/rząd/wiersz/g' file
```

Jeśli znak ukośnika, którego użyliśmy w tym przykładzie, występuje w poszukiwanym kluczu, można wybrać inny separator, taki jak znak „#” albo „|”.

```
sed 's#http://#
www.linux-magazine.net#http://#
www.linux-magazine.pl#g' url.html
```

Oczywiście, nie musimy dokonywać podstawienia w całym pliku; można wskazać jako adres poszczególne wiersze. Polecenie

```
sed 's/rząd/wiersz/g' file
```

wyszukuje i zamienia słowa tylko w pierwszym wierszu. Przełącznik *g* jest istotny w przypadku wielu wystąpień klucza w tym wierszu.

## Wiele poleceń na raz

Aby wykonać wiele czynności jednym wywołaniem edytora *sed*, należy wstawić przed poszczególnymi poleceniami znacznik *-e*. Na przykład następujące polecenie usuwa zawartość od piątego wiersza do końca pliku i wykonuje zamianę tekstu w pozostałej części pliku:

```
sed -e '5,$d' >
-e 's/KDE/Gnome/g' file
```

Zamiennie można rozdzielić polecenia średnikami i umieścić polecenia w nawiasach klamrowych. Wówczas poprzednie polecenie będzie wyglądać tak:

```
sed '{5,$d;s/KDE/Gnome/g}' file
```

## Czytanie i pisanie

Edytor *sed* ma również polecenia przeznaczone do czytania i pisania plików. Jeśli chcesz wstawić po trzecim wierszu plik zatytułowany *extras*, spróbuj użyć następującego polecenia

```
sed '3r extras' file
```

Polecenie *w* jest równie prostym sposobem na wydobywanie i zapisywanie zawartości:

```
sed '1,4w 1bis4.txt' file
```

zapisuje pierwsze cztery wiersze do nowego pliku *1bis4.txt*.

## Potęga wiersza poleceń

Jeśli zajdzie taka potrzeba, możesz grupować dowolną liczbę poleceń edytora *sed* w skrypcie, którym możesz potem „potraktować” określone pliki, używając opcji *-f*. Aby na przykład skasować drugi wiersz w pliku i dodać podany wiersz po wierszu czwartym, wpisz następujące polecenia w osobnych wierszach:

```
2d
4a\
Dodaj to po czwartym wierszu.
```

Polecenie *a* wymaga wstawienia znaku odwrotnego ukośnika i końca wiersza zaraz

po poleceniu (*4a*). Nowy wiersz tekstu, który ma zostać wstawiony, musi znajdować się w osobnym wierszu. Jeśli chcesz wstawić wiele wierszy tekstu, każdy z nich (poza ostatnim) musi być zakończony odwrotnym ukośnikiem:

```
4a\
To jest po czwartym wierszu.\
I to.\
I jeszcze ta ociupinka :)
```

Aby wstawić tekst przed danym wierszem zamiast po nim, wystarczy zamienić *a* na *i*:

```
1i\
To jest coś zupełnie nowego...
```

Później zachowaj te polecenia w pliku (z braku lepszej nazwy możesz nazwać go *script*), przekaz go programowi *sed* i zastosuj na jednym lub wielu plikach.

```
sed -f script file
```

## Bezpośrednie modyfikacje pliku

Jak wspomnieliśmy wcześniej, *sed* nie zmienia oryginalnego pliku, lecz wypisuje wyjście na *stdout*. Po upewnieniu się, że naprawdę chcemy wprowadzić zmiany, możemy przekierować wyjście. Operator *>* wysła wyniki do pliku, na przykład:

```
sed -f script file > newfile
```

Jeśli chcesz zmodyfikować oryginalny plik bez wykorzystywania tymczasowych rozwiązań, możesz użyć parametru *-i*. Po wywołaniu polecenia

```
sed -i -f tasks file
```

program *sed* nadpisuje pliki nowymi zmienionymi wersjami. Parametr umożliwia także utworzenie kopii bezpieczeństwa. Rozszerzenie dla kopii zapasowej określa się następująco:

```
sed -i.bak -f tasks file
```

Poza utworzeniem pliku *file* zawierającego zmiany, polecenie zachowuje oryginalny plik jako *file.bak* w tym samym katalogu.

## Kombinacje alpejskie

Program *ed* jest bardzo użyteczny w połączeniu z innymi programami wiersza pole-

ceń. Załóżmy, że mamy wiele plików ze spacjami i myślnikami w nazwach i chcielibyśmy zamienić te znaki na znak podkreślenia:

```
$ <B>ls -l *.mp3<B>
01 Saor_Free_News from >
Nowhere.mp3
02 Whirl-Y-Reel.mp3
...
```

W tym przypadku możesz utworzyć następujący skrypt zawierający reguły podstawień dla edytora *sed*:

```
s/ /_/g
s/-/_/g
```

Oczywiście, (choć nie ma to związku z naszym przykładem) możesz połączyć te dwa wiersze w pojedyncze wyrażenie regularne *s/[ -]/\_/g*. Uruchom skrypt na nazwach plików, żeby sprawdzić, czy reguły podstawień będą działać, kiedy ich naprawdę użyjemy. Jako że *sed* może czytać ze standardowego wejścia, za pomocą potoku można przekazać wyjście programu *ls* programowi *sed*:

```
$ ls -l *.mp3 | sed -f script
01_Saor_Free_News_>
from_Nowhere.mp3
02_Whirl_Y_Reel.mp3
```

Wszystko w porządku? Jeśli tak, dodajmy polecenie *mv*, aby zmienić nazwy plików. Dodatkowo dodajemy pętlę *for*, która da programowi *sed* bezpośredni dostęp do poszczególnych plików:

```
$ for i in *.mp3; do mv -v "$i" >
`echo $i | sed -f script`; done
01_Saor_Free_News from >
Nowhere.mp3' -> >
01_Saor_Free_News_from_>
Nowhere.mp3'
02_Whirl-Y-Reel.mp3' -> >
02_Whirl_Y_Reel.mp3'
```

Po polsku oznacza to: dla wszystkich plików z rozszerzeniem *\*.mp3* wykonaj następujące operacje: zmień oryginalną nazwę pliku na nazwę zwróconą przez program *sed* i wyświetl nazwy aktualnie zamienianych plików. Zmienną *\$i* trzeba umieścić w cudzysłowach, gdyż oryginalne nazwy plików mogą zawierać spacje. ■