Może się wydawać, że programy do edycji tekstu nie są konieczne w pracy administratora systemów. Nic jednak bardziej mylnego. Wyobraźmy sobie sytuację, że na serwerze zmieniamy kartę sieciową. No cóż awarie sprzętu się zdarzają. Po co nam do tego edytor tekstu? Do wymiany samej karty po nic, ale gdy mamy plik konfiguracyjny firewalla, w którym jest około 200 reguł zawierających łącznie około 100 wystąpień nazwy uszkodzonego interfejsu i musimy je zmienić to dobrze byłoby to zrobić jakoś automatycznie. A gdy mamy do dyspozycji tylko konsolę tekstową? W Unixie przewidziano odpowiednie narzędzia.

Edytor strumieniowy działa w oparciu o swego rodzaju skrypt, w którym opisane są modyfikacje jakie należy wykonać. Są to podstawowe operacje edycyjne jak: substytucja, usunięcie wiersza, wstawienie nowego wiersza z tekstem itp. Taki edytor czyta wskazany plik, a najczęściej jest to standardowe wejście, na którym są dane z innego programu w potoku, dokonuje zmian w odczytanym tekście i wypisuje wynik na standardowe wyjście, wiersz po wierszu. Domyślnie, jeśli w danym wierszu tekstu `sed` nie dokonuje żadnych zmian to wypisuje go takim jaki jest.

Edytor strumieniowy użyty w potoku z innymi programami zachowuje się jak filtr przepływającego przezeń tekstu. Wszystko dzieje się zupełnie automatycznie. Wcześniej, przed uruchomieniem odpowiedniego polecenia musi być tylko przygotowany skrypt opisujący zmiany.

Najbardziej znanym edytorem strumieniowym, rozpowszechnianym z Unixem, jest program `sed`. Jego nazwa to właśnie skrót od *stream editor*. Sam program ma tylko 3 opcje:

- `-e` : czyta polecenia edycyjne w formie specjalnego skryptu podanego w linii poleceń,
- `-f`: czyta polecenia edycyjne w formie specjalnego skryptu zapisanego we wskazanym pliku,
- `-n` : wstrzymuje wypisywanie wyników na standardowe wyjście.

Możliwości jednak tkwią nie w opcjach a w skrypcie. Przy jego tworzeniu obowiązują ścisłe reguły, to w zasadzie język programowania, dość prosty ale formalny.

Zacznijmy od prostego, ale obrazowego przykładu wykorzystującego najczęściej używane polecenie podstawienia (zamiany, substytucji):

```
sirius$ echo "hello world" | sed s/world/sed/
hello sed
```

Tutaj skrypt jest bardzo mały i kompaktowy, nie zawiera spacji ani znaków specjalnych, więc nie ma potrzeby go chronić. Znak `s` na początku oznacza polecenie podstawienia (od ang. *substitute*). Znaki slash `/` ograniczają wyrażenie regularne, które jest dopasowywane i tekst na jaki zamienić znalezioną frazę. Tutaj akurat wyrażenie jest trywialne, nie zawiera żadnych znaków specjalnych. W miejscu znaków slash / możemy użyć innego znaku, jaki nie występuje w szukanym wyrażeniu i tekście do podstawienia, na przykład:

```
sirius$ echo "hello world" | sed sxworldxsedx
hello sed
```

Użyty tutaj znak `x` trochę się zlewa i czyni polecenie nieczytelnym, ale jest to dopuszczalne bo `x` nigdzie nie występuje. W sytuacji, gdy modyfikujemy ścieżki zawierające znak slash `/`, wygodnie jest użyć innego znaku, na przykład `%`:

```
sirius$ echo /tmp/moj/katalog/z/plikami | sed s%moj/katalog%podkatalog%
/tmp/podkatalog/z/plikami
```

Polecenie substytucji może zawierać dodatkowe opcje:

- `i` : ignorowanie wielkości liter,
- `g` : przeszukiwanie globalne.

Zobaczmy jak one działają w praktyce.

```
sirius$ echo "hello world" | sed s/WORLD/sed/
hello world
```

Domyślnie sed odróżnia małe i wielkie litery. Dlatego powyżej nie zostały wykonane żadne zmiany. Dodanie opcji `i` zmienia to zachowanie:

```
sirius$ echo "hello world" | sed s/WORLD/sed/i
hello sed

sirius$ echo "hello world a wonderful world" | sed s/world/sed/
hello sed a wonderful world
```

Normalnie sed dopasowuje wzorzec do pierwszego znalezienia w wierszu. Aby zamienić wszystkie wystąpienia wzorca w wierszu należy użyć opcji `g`:

```
sirius$ echo "hello world a wonderful world" | sed s/world/sed/g
hello sed a wonderful sed
```

Gdy chcemy wykonać kilka poleceń to należy przed każdym użyć opcji `-e`:

```
sirius$ echo "hello world" | sed -e s/h/H/ -e s/world/sed/
Hello sed
```

To samo uzyskamy dwa razy wywołując program sed, ale to mniej efektywne:

```
sirius$  echo "hello world" | sed s/h/H/ | sed s/world/sed/
Hello sed
```

Polecenia do wykonania przez `sed` można zamieścić w pliku. Jest to szczególnie wygodne, gdy tych poleceń jest więcej albo, gdy korzystamy z nich wielokrotnie.

```
sirius$ echo s/world/sed/ > /tmp/sed.cmd
sirius$ echo "hello world" | sed -f /tmp/sed.cmd
hello sed
```

Tutaj zapisaliśmy nasze polecenie w pliku `/tmp/sed.cmd`. Aby `sed` wykonał skrypt z pliku należy go wskazać za pomocą opcji `-f`.

Zobaczmy jakie inne operacje można wykonać za pomocą `sed` na pliku tekstowym. Aby dokładnie omówić wszystkie możliwości programu sed należałoby poświęcić mu nie część wykładu, ale kilka wykładów. Ograniczymy się zatem do podstaw. W sieci można znaleźć mnóstwo tutoriali poświęconych temu programowi.

Do testowania będziemy potrzebować jakiś plik tekstowy. Powiedzmy, że w bieżącym katalogu mamy plik `ipf.conf` zawierający reguły firewalla (to tylko fragment rzeczywistej konfiguracji):

```
# Reliable networks
pass in quick from pool/reliable
# Intruders
block return-icmp(port-unr) in quick from pool/intruders
# Spoofing
```

```
block in quick from pool/spoofing
# Rules
pass in quick on igb0 proto tcp from 192.168.0.2/32 to any flags S keep state
pass in quick on igb0 proto udp from 192.168.0.2/32 to any keep state
block in log quick on igb0 proto tcp from any to any port = 25
block in on igb2 all
pass in quick on igb2 proto udp from 0.0.0.0/32 port = 68 to 255.255.255.255/32 port = 67
pass in quick on igb2 proto udp from 172.16.0.0/32 port = 68 to 172.16.0.1/32 port = 67
pass in quick on igb2 proto icmp from 172.16.0.0/24 to 172.16.0.1/32 icmp-type 8 keep state
pass in quick on igb2 proto tcp from 172.16.0.0/24 to 172.16.0.1/32 port = 53 flags S keep state
pass in quick on igb2 proto udp from 172.16.0.0/24 to 172.16.0.1/32 port = 53 keep state
pass in quick on igb2 proto tcp from 172.16.0.0/24 to 172.16.0.1/32 port = 80 flags S keep state
pass in quick on igb2 proto tcp from 172.16.0.0/24 to 172.16.0.1/32 port = 443 flags S keep state
```

Zachęcam do samodzielnego testowania.

Polecenia `sed` to jednoliterowe skróty. Umieszcza się je po podaniu adresu lub zakresu adresów Polecenie podstawienia jest wyjątkiem. Adresem jest numer wiersza lub wzorzec. Wzorcem jest podstawowe wyrażenie regularne, a zakres tworzymy rozdzielając adresy przecinkiem. Wzorzec ograniczony jest zwykle znakiem slash `/`, ale można użyć innego znaku, tak jak dla polecenia podstawienia.

Możemy używać programu `sed` zamiast `grep`. Opcja `-n` wyłącza wypisywanie wyników na standardowe wyjście, a polecenie `p` (od ang. *print*) wymusza wypisanie wyniku. Polecenie:

```
sed -n /state/p ipf.conf
```

wypisze tylko wiersze zawierające frazę `state` i jest równoważne z:

```
fgrep state ipf.conf
```

natomiast polecenie:

```
sed -n /^#/p ipf.conf
```

wypisze tylko komentarze i jest równoważne z:

```
grep ^# ipf.conf
```

Aby usunąć komentarze możemy użyć polecenia:

```
sed /^#/d ipf.conf
```

Tutaj literka `d` to polecenie usuwania (od ang. *delete*) całego wiersza, który zawiera pasujący do wzorca ciąg. Jak to zrobić za pomocą `grep`?

Wypiszmy teraz tylko te linie naszego pliku, które nie są komentarzami i nie zawierają cyfr. Wykonujemy tutaj dwie operacje. Albo umieszczamy je w pliku w osobnych wierszach, albo używamy opcji `-e` dwukrotnie:

```
sed -e /^#/d -e /[0-9]/d ipf.conf
```

A co jeśli chcemy usunąć komentarze i wiersze nie zawierające cyfr?

```
sed -e /^#/d -e '/[0-9]/!d' ipf.conf
```

Tutaj potrzebujemy ochronić drugie polecenie z uwagi na znak wykrzyknika `!` , który neguje polecenie, to znaczy nie usuwaj wierszy zawierających cyfrę.

Polecenie `a` (od ang. *append*) dopisuje tekst po wierszu o wskazanym adresie lub po każdym wierszu z określonego zakresu. Na przykład:

```
sed '/^#/a # These are the rules' ipf.conf
```

dopisze tekst `# These are the rules` po każdym komentarzu.

Polecenie `i` (od ang. *insert*) wstawia tekst przed wierszem o wskazanym adresie lub przed każdym wierszem z określonego zakresu. Na przykład:

```
sed '/^bloc/a # This is a blocking rule' ipf.conf
```

wstawi `# This is a blocking rule` przed każdym wystąpieniem frazy `block` na początku wiersza.

Polecenie `c` (od ang. *change*) zamienia wiersz określony adresem lub wiersze określone zakresem na tekst podany tekst. Na przykład:

```
sed '/keep state/c # Here was a keep state rule' ipf.conf
```

zamieni wszystkie reguły keep state na komentarz.

Zobaczmy jak używa się zakresów. Polecenie:

```
sed /Reliable/,/Rules/d ipf.conf
```

usunie wiersze od wiersza zawierającego frazę `Reliable` do wiersza zawierającego frazę `Rules` włącznie. Na marginesie, nie używamy w tym zaklęciu znaków specjalnych shella więc nie musimy stosować ochrony.

Aby nie powstało mylne wrażenie, że zakresów używa się tylko do usuwania wypiszmy wiersze, które usuwaliśmy poprzednio:

```
sed -n /Reliable/,/Rules/p ipf.conf
```

Zwróćmy uwagę, że polecenie `p` ma sens wyłącznie razem z opcją `-n`.

Zamiast wzorców można używać numerów linii. Aby wypisać wiersz 4 użyjemy polecenia:

```
sed -n '4p' ipf.conf
```

Aby przed wierszem 7 wstawić tekst użyjemy:

```
sed '7i # No more pools' ipf.conf
```

Numery linii zamiast wzorców możemy stosować także do określania zakresów. Aby usunąć wiersze od 1 do 11 z naszego pliku możemy użyć polecenia:

```
sed 1,11d ipf.conf
```

Można także mieszać numery ze wzorcami. Polecenie:

```
sed -n '3,/^block/p' ipf.conf
```

wypisze wiersze od 3 do wiersza zawierającego ciąg `block` na początku.

Wracając do przykładu podanego na samym początku uzasadniającego sens edytorów strumieniowych zamieńmy w naszym pliku wszystkie wystąpienia interfejsu `igb2` na `e1000g0`:

```
sed s/igb2/e1000g0/g ipf.conf
```

Wynik jednak zostanie tylko na ekranie. Aby na trwale dokonać zmian w naszym pliku musimy zapisać wynik i skopiować go na nasz plik:

```
sed s/igb2/e1000g0/g ipf.conf > /tmp/x
cp /tmp/x ipf.conf
rm /tmp/x
```

Plik `/tmp/x` jest tymczasowy i po wszystkim należy go usunąć. Można też:

```
sed s/igb2/e1000g0/g ipf.conf > /tmp/x
mv /tmp/x ipf.conf
```

ale możemy w ten sposób zmienić własność i uprawnienia pliku `ipf.conf`. Kopiowanie przez `cp` je zachowa.