Dlaczego warto używać zsh

Mikołaj Słupiński

3 stycznia 2018

1 Wstęp

Systemy z rodziny UNIX pozwalają użytkowniowi na swobodę konfiguracji. Umożliwia nam to między innymi zmianę powłoki systemu. Bez wątpienia, najpopularniejszą obecnie powłoką jest instalowany domyślnie na większości dystrybucji linuksa bash (ang. *Bourne-Again Shell*), jednakże świadomy użytownik systemu operacyjnego powinien przynajmniej spróbować innych, konkurencyjnych rozwiązań. Z tego powodu postaram się przybliżyć cechy zsh (*Z Shell*), które wyróżniają ją na tle innych produktów.

2 Ciekawe funkcje zsh

2.1 Automatyczne uzupełnianie

Funkcjonalność basha w tej sferze jest mocno ograniczona. Owszem, pozwala on na uzupełnianie klawiszem tabulacji nazw ścieżek, czy plików, jednakże zsh oferuje bardziej zaawansowane rozwiązania. Klawisz tabulacji pozwala nam na uzupełnianie opcji menedżera pakietów czy programu git. Ponadto, po wpisaniu fragmentu polecenia, wciśnięcie strzałki w górę spowoduje odnalezienie ostatnio użytego polecenia rozpoczynającego się daną sekwencją znaków.

2.2 Autokorekta

Wpisanie w powłoce `setopt correct` spowoduje włączenie opcji autokorekty. W momencie kiedy wpisując polecenie popełnimy literówkę, np.

```
nerd911@nerd911-laptop  ~  exti
```

Wyświetli nam się komunikat zbliżony do zamieszczonego poniżej:

```
zsh: correct 'exti' to 'exit' [nyae]?
```

co pozwala nam szybko poprawić literówkę.

2.3 Rozwijanie ścieżek

Jedną z bardziej przydatnych opcji jest rozwijanie ścieżek przez zsh. Wpisanie `cd /u/lo/b`, a następnie wciśnięcie przycisku *tab* spowoduje rozwinięcie polecenia do `cd /usr/local/bin`, co zaoszczędza nam pisania.

2.4 Globowanie

Glob to krótkie wyrażenie, które pozwala nam na wybranie dużej ilości plików, za pomocą jednego zapytania, które można uzupełnić o wyrażenia regularne. Proste przykłady:

```
# Wypisz kazdy plik znajdujacy sie dokladnie w katalog moj_katalog
ls moj_katalog

# Wypisz kazdy plik znajdujacy sie dokladnie jeden poziom
# pod katalogiem moj_katalog
ls moj_katalog/*
```

```
# Wypisz kazdy plik znajdujacy sie dokladnie dwa poziomy
# pod katalogiem moj_katalog
ls moj_katalog/*/*

# Wypisz kazdy plik znajdujacy sie w katalogu moj_katalog
# i jego podkatalogach
ls moj_katalog/**/*

# Wypisz kazdy plik o rozszerzeniu txt znajdujacy sie w
# katalogu moj_katalog i jego podkatalogach
ls moj_katalog/**/*.txt
```

Możemy również wykorzystać to np. do iterowania po podkatalogach:

```
for katalog in moj_katalog/dane/*/*; do
    # utworz plik w kazdym podkatalogu
done
```

2.5 Kwalifikatory

Kwalifikatory pozwalają nam na filtrowania wyszukiwanych przez nas plików. Są to proste wyrażenia, które umieszczamy na końcu zapytań w okrągłych nawiasach. Kilka prostych przykładów:

```
# Wypisz tylko foldery
print -l moj_katalog/**/*(/)

# Wypisz tylko zwykle pliki
print -l moj_katalog/**/*(.)

# Pokaz tylko puste pliki
ls -l moj_katalog/**/*(L0)

# Pokaz tylko pliki wieksze niz 3 KB
ls -l moj_katalog/**/*(Lk+3)

# Pokaz pliki zmodyfikowane w ciagu ostatniej godziny
print -l moj_katalog/**/*(mh-1)

# Posortuj pliki od w kolejnosci czasu od ostatniej modyfikacji
# i wypisz 3 pierwsze rekordy
ls -l moj_katalog/**/*(om[1,3])
```

2.6 Modyfikatory

Modyfikatory są podobne w użyciu do kwalifikatorów. Pozwalają nam one na bardziej zaawansowane operacje na ścieżkach do plików. Aby odróżnić je od kwalifikatorów są one poprzedzone znakiem dwukropka.

```
# Zwykly glob
print -l moj_katalog/dane/podkatalog1/podkatalog2/*.txt

# Wypisz nazwe pliku (t pochodzi od tail)
print -l moj_katalog/dane/podkatalog1/podkatalog2/*.txt(:t)

# Wypisz nazwe pliku bez rozszerzenia (r pochodzi od remove_extension)
print -l moj_katalog/dane/podkatalog1/podkatalog2/*.txt(:t:r)

# Wypisz rozszerzenie
print -l moj_katalog/dane/podkatalog1/podkatalog2/*.txt(:e)
```

```
# Wypisz katalog nadrzedny pliku (h od head)
print -l moj_katalog/dane/podkatalog1/podkatalog2/*.txt(:h)

# Wypisz katalog nadrzedny katalogu nadrzednego
print -l moj_katalog/dane/podkatalog1/podkatalog2/*.txt(:h:h)

# Mozna laczyc kwalifikatory z modyfikatorami
# Wypisz karalog nadrzedny pierwszego pliku
print -l moj_katalog/dane/podkatalog1/podkatalog2/*.txt([1]:h)
```

3 Podsumowanie

Oczywiście nie są to wszystkie funkcjonalności zsh, jednakże opisanie wszystkich zastosowań powłoki zajęłoby zdecydowanie więcej, niż krótki artykuł. Czasem warto otworzyć się na zmiany i spróbować innych rozwiązań niż te dostarczone razem z systemem. Oczywiście wymaga to zmierzenia się z dokumentacją, jednak jak pokazuję powyższe przykłady, niejednokrotnie warto.