THEORETICAL COMPUTER SCIENCE
Jagiellonian University

xargs

Polecenie `xargs` służy do konstruowania listy argumentów ze standardowego wejścia dla innych poleceń i wykonywania tychże poleceń. Brzmi to dość abstrakcyjnie i może enigmatycznie ale kilka zastosowań polecenia `xargs` jest uważanych za jedne z najbardziej użytecznych sekwencji komend w Linuksie.

Przykład. Skasuj wszystkie pliki z rozszerzeniem `.c` w bieżącym katalogu. Oczywiście możemy zrobić to tak:

```
rm *.c
```

Możemy, też tak:

```
find . -name "*.c" | xargs rm
```

To jednak nie zadziała gdy mamy pliki z białymi znakami w nazwie. Możemy to poprawić tak, aby nie było komunikatu błędu:

```
find . -name '*.c' | xargs rm -f
```

Ale plik z białymi znakami w nazwie dalej pozostanie nieskasowany. Poprawnie zadziała dopiero następująca sekwencja:

```
find . -name '*.c' -print0 | xargs -0 rm
```

Wprawdzie polecenie `find -exec rm` pozwala nam usuwać znalezione pliki, to następuje to wolniej niż skorzystanie z `xargs`. Jest tak, ponieważ `find` uruchamia dla każdego pliku pojedynczo plecenie `rm`. Natomiast `xargs` przekazuje tyle argumentów ile polecenie maksymalnie może przyjąć. W przypadku `rm` jest to na tyle duża liczba, że znacząco zwiększa wydajność usuwania plików.

Przykład. Znajdź wszystkie pliki z rozszerzeniem `.c` zawierające sekwencję *stdlib.h*. W tym wypadku nie jest oczywiste jak to wykonać bez komendy `xargs`. Użycie `xargs` zaś jest proste:

```
find . -name '*.c' | xargs grep 'stdlib.h'
```

Przykład. Dla każdego pliku w bieżącym katalogu o rozszerzeniu `.txt` wydrukuj jedną linię na standardowe wyjście zawierającą liczbę linii w tym pliku i jego nazwę.

```
ls -1 *.txt | xargs -n 1 wc -l
```

Zwróć uwagę na opcję `xargs -n 1` w powyższej linii. Za co ona odpowiada? Jaka jest różnica między wywołaniem z ta opcją i bez tej opcji?

Przykład. Przesuń wszystkie pliki o rozszerzeniu `.bak` do katalogu `~/old.files`.

```
find . -name "*.bak" -print0 | xargs -0 -I {} mv {} ~/old.files
```

```
find . -name "*.bak" -print0 | xargs -0 -I file mv file ~/old.files
```

Zwróć uwagę na opcję `-I`. Po niej podawany jest znacznik, który ma być interpretowany w poleceniu jako miejsce gdzie `xargs` ma wkleić dane otrzymane na standardowym wejściu. Domyślnie `xargs` wstawia te dane na końcu tworzonego polecenia (patrz poprzednie przykłady). Jednak polecenie `mv` wymaga aby najpierw była nazwa pliku (bądź plików) które mają być przeniesione, a na końcu ma być miejsce przeniesienia. Powyższe dwie linie różnią się jedynie ciągiem znaków podanym jako wspomniany znacznik.

Przykład. W każdej linii pliku „test.in" znajduje się jedno słowo. Utwórz pliki o nazwach takich jak słowa z „test.in" Niestety polecenie `touch` nie wczytuje danych ze standardowego wejścia, a jedynie jako argumenty. Więc jedynym sposobem jest użycie `xargs`:

```
cat test.in | xargs touch
```

Tak jak wcześniej omawialiśmy powyższa sekwencja nie zadziała dla nazw z białymi znakami.

Przykład.

```
cat plik | xargs -n 1 -I {} sh -c "mkdir {}; touch {}/{}"
```

Dla każdego słowa w pliku `plik` tworzy katalog o tej nazwie i pusty plik w tym katalogu o tej samej nazwie. Zwróć uwagę, iż `xargs` wywołuje tutaj nową instancje powłoki poleceń (poprzez polecenie `sh`). Dzięki temu możemy wykonać więcej niż jedno polecenie używając argumentów przekazanych przez `xargs`.

Przydatne opcje `find`

(i) `find . -name '*.c' -maxdepth 1` — drukuje nazwy plików i katalogów o nazwie zakończonej `.c` w aktualnym katalogu ale nie w jego podkatalogach,
(ii) `find . -name '*.c' -type f` — drukuje nazwy plików o nazwie zakończonej `.c` w aktualnym katalogu i podkatalogach,
(iii) `find . -name '*.c' - type f -size 0` — drukuje nazwy pustych plików w aktualnym katalogu i podkatalogach