Tak jak było powiedziane na pierwszym wykładzie, ogromne możliwości systemu Unix biorą się stąd, że mamy w nim mnóstwo drobnych programów narzędziowych. Każdy z nich wykonuje jedno zadanie, ale można je łączyć ze sobą przy pomocy *potoków*, tak aby zrealizować czasem bardzo skomplikowane zadania.

Jak wiemy, każdy proces w systemie ma przypisane 3 standardowe deskryptory: wejście, wyjście i błędy. Potok tworzymy pisząc znak | pomiędzy sąsiednimi poleceniami. W potoku może wystąpić kilka poleceń, a więc kilka znaków |. Wszystko zależy od tego co chcemy uzyskać. Potok działa w bardzo prosty sposób, ale daje to ogromne możliwości. Standardowe wyjście programu po lewej stronie | zostaje skojarzone ze standardowym wyjściem programu z prawej strony |. Inaczej mówiąc to co program z lewej strony normalnie wypisałby na ekran trafia jako dane wejściowe do drugiego programu. Proste prawda? Ale ile można w ten sposób zdziałać?

Przykład. Program `cat` służy do łączenia wielu plików w jeden i jeśli tego nie zmienimy wynik trafia na ekran. Tak więc polecenie

```
cat a.txt b.txt c.txt
```

wypisze zawartość plików `a.txt`, `b.txt` oraz `c.txt` na ekran , tak jakby to był jeden plik. Program `wc` (word count) zlicza znaki (opcja `-c`) , wiersze (opcja `-l`) oraz słowa (opcja `-w`). Jak policzyć ile razem wierszy zawierają nasze 3 pliki? Proste, wystarczy przekazać to co wyprodukuje `cat` do `wc`:

```
cat a.txt b.txt c.txt | wc -l
```

Gdybyśmy chcieli policzyć ile niepustych linii kodu zawierają wszystkie programy napisane w C to wtedy:

```
cat *.c | grep -v ^$ | wc -l
```

Pierwsze polecenie zlepi wszystkie pliki z `.c` na końcu nazwy (w Unixie nie mówi się o rozszerzeniu pliku), drugie `grep` odfiltruje niepuste linie (co znaczą te dziwne znaczki stanie się jasne jak poznamy wyrażenia regularne później), trzecie polecenie już znamy. Jeśli jednak cześć kodu jest w plikach w podkatalogach w bieżącym katalogu to powyższe polecenie nie uwzględnia plików w podkatalogach. Wtedy z pomocą przychodzi `find`:

```
find . -name "*.c" -exec cat {} \; | grep -v ^$ | wc -l
```

Program `find` może się wydawać trudny dla początkujących, ale jest niezbędnikiem w rękach administratora Unixa. Kropka za `find` to katalog w systemie plików od którego `find` ma zacząć poszukiwania wgłąb. Kropka oznacza bieżący katalog, ten w którym jesteśmy (czasem kropka się przydaje). Opcja `-name` powoduje odszukiwanie tylko tych plików, które pasują do podanej nazwy (znaki `*`, `?` , `[`, `]` itp. trzeba chronić w `""`). Opcja `-exec` wykonuje dowolny program na każdym ze znalezionych plików (mówiąc plik mam na myśli nie tylko zwykłe pliki, ale katalogi, linki, sokety, urządzenia itd.). Tutaj wykonywane jest `cat`. Nawiasy `{}` oznaczają, że w to miejsce ma być podstawiona ścieżka do znalezionego pliku, natomiast `\;` to ochroniony znakiem `\` średnik kończący polecenie po `-exec`. Dalej w tym potoku mamy to samo co poprzednio.

Na zakończenie opisu potoków podam przykład, który pokazuje ich moc:

```
fgrep '08/Nov/2020' access.log | awk '{print $1}' | sort | uniq -c | sort -n
-r | head -15
```

Polecenie to wyświetla 15 adresów IP, z których 8 listopada 2020 było najwięcej połączeń do serwera HTTP, którym administrujemy. Plik `access.log` to dziennik serwera HTTP, gdzie spisane jest każde wejście jako osobna linia. W każdej linii tego pliku na początku jest adres IP. Tak ona może wyglądać:

```
185.191.171.44 - - [08/Nov/2020:22:17:59 +0100] "GET
/pl/pracownicy/pracownik.php?cID=158&s=zajecia HTTP/1.1" 200 6781 "-"
"Mozilla/5.0 (compatible; SemrushBot/6~bl;
+http://www.semrush.com/bot.html)"
```

Nasze polecenie najpierw za pomocą `fgrep` odfiltrowuje z `access.log` te linie, gdzie jest interesująca nas data w odpowiednim formacie. Program `awk` wyławia z takiej linii tylko pierwsze pole. Program `sort` sortuje znalezione adresy IP, a `uniq` eliminuje wielokrotne, sąsiednie (dlatego sortowanie) wystąpienia jednego IP, podaje też liczbę wystąpień. Drugie `sort` sortuje malejąco według wartości liczbowych podanych przez `uniq`. Ostatecznie `head` wypisze tylko pierwszych 15 wystąpień.

Zachęcam do eksperymentowania z tym poleceniem na załączonym pliku dziennika. Proszę wykonywać kawałek potoku po kawałku i patrzeć co robią poszczególne jego składowe.