3. Narzędzia systemowe

Podstawowe pojęcia:

1. Standardowe kanały komunikacyjne (standardowe strumienie danych)

   Programy, które przetwarzają jakieś dane oraz wypisują rezultaty tych przetworzeń, potrzebują kanałów komunikacyjnych, aby pobrać dane wejściowe i przesłać gdzieś wyniki swojej pracy. Dostępne są trzy standardowe kanały komunikacyjne:

   - Standardowe wejście (*stdin*) (klawiatura, plik lub wyjście innego polecenia)–służy do pobierania danych,
   - Standardowe wyjście (*stdout*) (ekran, plik lub wejście innego polecenia) – służy do odbierania danych,
   - Standardowy strumień błędów lub standardowe wyjście błędów (*stderr*) (ekran, plik) –służy do odbierania komunikatów o błędach.

2. Wyrażenia regularne

   Wyrażenia regularne to w informatyce teoretycznej ciągi znaków pozwalające opisywać języki regularne. W praktyce znalazły bardzo szerokie zastosowanie, pozwalają bowiem w łatwy sposób opisywać wzorce tekstu, natomiast istniejące algorytmy w efektywny sposób określają, czy podany ciąg znaków pasuje do wzorca lub wyszukują w tekście wystąpienia wzorca. Wyrażeni regularne są tworzone za pomocą liter cyfr i specjalnych znaków, których działanie przypomina działanie operatorów. Dzięki tym wyrażeniom łatwiej jest filtrować informacje w plikach. Wyrażenia regularne stanowią integralną część narzędzi systemowych.

   Proste wyrażenia regularne występują albo samodzielnie, albo za pomocą operatorów są łączone z innymi wyrażeniami regularnymi – złożone wyrażenia regularne.

Operatory wyrażeń regularnych

zapis	opcje
`c`	Dowolny znak, który nie jest znakiem specjalnym
`.`	Dopasowuje jeden dowolny znak (odpowiednik ? w metaznakach generujących nazwy plików)
`$`	Jeśli jest to ostatni znak wyrażenia regularnego to odpowiadający mu ciąg znaków musi wystąpić na końcu linii
`^`	Jeśli jest to pierwszy znak wyrażenia regularnego to odpowiadający mu ciąg znaków musi wystąpić na początku linii
`*`	Zero lub więcej wystąpień znaku poprzedzającego operator (inaczej niż dla metaznaków)
`\`	Użyty przed znakiem specjalnym pozbawia ten znak znaczenia, ale występując przed dowolnym innym znakiem oznacza sam siebie
`[lista]`	Oznacza jeden ze znaków znajdujących się na liście (określa zakres np.:[0-7] lub serię znaków np.: [absdh] lub oba razem np.: [absdh0-7]
`[^lista]`	Oznacza jeden znak, ale nie znajdujący się na liście
`\\`	Puste wyrażenie regularne, które przyjmuje wartość ostatniego wyrażenia regularnego
`\{m\}`	Dokładnie m wystąpień znaku lub znaków poprzedzających wyrażenie
`\{m,\}`	Przynajmniej m wystąpień znaku lub znaków poprzedzających wyrażenie
`\{m,n\}`	Od m do n wystąpień znaku lub znaków poprzedzających wyrażenie
`\(…\)`	Zapisanie w pamięci wzorca zamkniętego pomiędzy „\(„a”\)”. Odwołanie do niego następuje przez znaki od \1 do \9

Przykłady

1. Symbol .

Oznacza dowolny pojedynczy znak ASCII, używa się w celu dopasowania dowolnego, pojedynczego znaku niezależnie od tego jaki to będzie znak.
Trzyliterowy napis `1.s` może oznaczać słowa *las*, *los* ale również *laser* czy *leszczyna* oraz ciągi bez sensu jak *lfs* czy *lgs*.

2. Symbol **$**

   Znak ten umieszczony po wyrażeniu regularnym używany jest do dopasowania wzorca do końca wiersza. Wyrażenie **koniec$** dopasowane zostanie pod warunkiem, że ciąg znaków *koniec* wystąpi na końcu linii w innym wypadku ciąg znaków *koniec* zostanie odrzucony.
3. Symbol **^**

   Zastosowanie takie samo jak symbol **$**, tylko oznacza początek wiersza. W połączeniu z **$**: **^$** oznacza pustą linię a np.: **^ala$** oznacza linię zawierającą tylko ciąg *ala*.
4. Symbol ***

   Jest używany do dopasowania zera lub większej liczby znaków wystąpień poprzedzającego znaku czyli powtórzeń. I tak wyrażenie **go*d** może oznaczać *gd*, *god*, *good* itp. ale jeśli napiszemy **g.*d** to zapis może mieć dowolny ciąg znaków pomiędzy *g* a *d* bo **.*** właśnie to oznacza. Jeśli tylko umieścimy wyrażenie **.*** będzie ono odpowiadało całemu wierszowi, ponieważ wyrażenia regularne w przypadku niejednoznaczności są dopasowywane do najdłuższego napisu. A więc wyrażenie **^ *$** będzie odpowiadał wiersz składający się z samych spacji.
5. Symbol **[ ] [^]**

   Oznacza definicję zbioru znaków w wyrażeniu regularnym. Każdy znak w nawiasach znajduje się w tym zbiorze, a wystąpienie choć jednego znaku w badanym ciągu jest uważane za dopasowane. Jeśli zamiast **l.s** napiszemy **l[aoi]s** to otrzymamy *las*, *los*, *lis* ale również *lisy*, *losy* czy *liseczk*i jeśli jednak umieścimy znak spacji po *s* to zawężimy nasze poszukiwania do trzyliterowych wyrażeń.

   Jeśli chcemy znaleźć ciąg znaków rozpoczynający się od samogłoski to **^[aeiou]***, aby zaczynał się od cyfry **^[0-9]**, a wzorzec dla dowolnej liczby całkowitej poprzedzonej znakiem dodawania i odejmowania wygląda następująco **[-+][0-9][0-9]***, jeśli chcemy wykluczyć jakiś znak to korzystamy z **[^]** i tak wyrażenie **[^A-Za-z]** odpowiada wszystkim znakom, które nie są literami. A **^[^0-9].*$** oznacza linie zaczynające się dowolnym znakiem oprócz liczb.
6. Symbole typu **\{m,n\}**

   Aby kontrolować ilość powtórzeń danego znaku w tekście należy skorzystać z symbolów typu **\{m,n\}**. I tak **A\{2\}** oznacza *AA*, ale nie *A* czy *AAA*. **A\{2,\}** oznacza *AA*, *AAA*, *AAAA*, itd. ale nie *A*. **A\{2,4\}** oznacza *AA*, *AAA*, *AAAA* ale nie *A* czy *AAAAA*. Możemy tworzyć bardziej rozbudowane wyrażenia np.: nasze wcześniejsze wyrażenie możemy zastąpić **[-+]\{0,1\}[0-9][0-9]*** co oznacza, że znak – i + może wystąpić raz lub wcale.
7. Symbole typu **\(...\)**

   Symbole typu **\(...\)** są to tzw. podwyrażenia. Dla pojedynczego wiersza może zostać zdefiniowanych max. 9 podwyrażeń. Zamiast kropek definiuje się podwyrażenie a następnie można się do niego odwołać przez **\numer_podwyrażenia**. I tak **ding \(dong\) \1** oznacza *ding dong dong* (*dong* to podwyrażenie) czy **\(tik\) \(tak\) \1 \2** to *tik tak tik tak*.

Polecenia grep, egrep, fgrep

Polecenie grep wyświetla linie zawierające podany wzorzec (słowo lub ciąg znaków). Wzorzec może zawierać pewne znaki specjalne ujęte w ` `, inaczej nie zostaną zinterpretowane jako wzorzec. jeśli przeszukiwane jest kilka plików, o przed każdą znalezioną linią jest wyświetlana nazwa pliku. jeśli nie podano nazw plików pobierane są ze standardowego wejścia, co pozwala to polecenie wykorzystać w potokach. do tej rodziny poleceń nalezy również egrep i fgrep. Polecenie egrep używa pełych wyrażeń regularnych i może szukać wielu wzorców, natomiast polecenie fgrep może szukać wielu wzorców ale tylko zwykłych słów lub ciągów znaków (operatory wyrażeń regularnych są tutaj interpretowane dosłownie).

```
grep [-opcje] ograniczone_wyrażenie_regularne plik
egrep [-opcje] pełne_wyrażenie_regularne plik
```

fgrep [-opcje] wzorzec plik

-b – poprzedza każdy wyszukany wiersz numerem bloku, w którym się on znajduje (pierwszy blok ma numer 0)
-c – Wyświetla tylko liczbę wierszy zawierających wzorzec. Jeśli plików jest więcej niż jeden wyświetla tę liczbę dla każdego pliku osobno
-e wzorzec – opcji tej używa się jeśli wzorzec poprzedzony jest znakiem -
-f – pobiera poszukiwane ciągi znaków (napisy) z pliku
-h – wypisywane są zgodne wiersze, ale nie są wypisywane nazwy plików (operacja przeciwna do –l)
-i – wielkie i małe litery nie są rozróżniane w czasie porównywania z wzorcem
-l – wyświetlane są nazwy plików, ale nie są wypisywane zgodne wiersze (operacja przeciwna do –h)
-n –poprzedza każdą linię jej numerem (linia pierwsza ma numer 1)
-s – wyłącza wyświetlanie komunikatów o błędach
-v – wyświetla linie niezawierające wzorca
-x – wyświetla tylko te linie w całości pasujące do wzorca

Wyrażenie grep nie dopuszcza podwyrażeń w zamian oferuje następujące operatory:

+ - jedno lub więcej wystąpień wyrażenia regularnego poprzedzającego symbol,
? – zero lub jedno wystąpienie wyrażenia regularnego poprzedzającego symbol,
| - Wyrażenie regularne wyszczególnione przed lub po symbolu,
( ) – przypisywanie identycznych fragmentów do zamkniętych w nawiasach grup wyrażeń regularnych.

Przykłady

8. Operator **+** jest podobny do *****, z tym, że nie pozwala, aby zero wystąpień spełniało warunek wyrażenia. Poprzedzający znak musi wystąpić przynajmniej raz. I tak **go+d** oznacza *god*, *good*, *goood* itd. ale nie *gd*. Stosując ten operator możemy skrócić większość wyrażeń opisywanych przez *****, np.: wyrażenie opisujące dwa dowolne słowa **[A-Za-z][A-Za-z]*** możemy zastąpić przez **[A-Za-z]+** lub wyrażenie **[-+]\{0,1\}[0-9][0-9]*** na **[-+]\{0,1\}[0-9]+** lub jeszcze bardzie uprościć **[-+]?[0-9]+**, gdzie symbol **?** oznacza zero lub jedno wystąpienie poprzedzającego znaku. Wrażenie regularne na liczbę całkowitą lub rzeczywistą wyglądałoby tak: **[-+]?[0-9]+\.?[0-9]+**. Operator **|** lub znak nowej linii oznacza, że zostanie dopasowane jedno z dwóch rozdzielnych wyrażeń regularnych. Tak więc: **ogeń|woda** pozwala szukać linii zawierającej w sobie jedno lub drugie słowo. Nawiasy pozwalają grupować wyrażenia regularne. Także znaki ***,+, ?, |** mogą się odnosić do określonych części wyrażenia, np.: **(go)?** oznacza *go* i *gogo*, ale **go?** to już *go* i *goo*.

Ćwiczenia

1. Utwórz plik *test.txt* i wpisz do niego następujący tekst:

Ala ma kota.
Kot to zwierzatko, ktore kocha Ala.
Mama kupila nowa ksiazke Ali.
Karolina i Ala to dwie przyjaciolki.
Karolina ma brata Karola.
Karol jest starszy od Karoliny o 2 lata.
Ala kocha sie w Karolu.
Karol ma 17 lat, a Ala ma 16 lat.

2. W pliku *test.txt* odszukaj:
   - linie nie zawierające słowa *Ala* **gerp –v Ala test.txt**
   - linie zaczynające się od słów *Ala* i *ala* **gerp '^[Aa]la' test.txt**
   - numer linii z frazą *Karolina ma brata* **gerp –n 'Karolina ma brata' test.txt**

- numery linii z wyrazami z pliku *wz* (Karolina, Mama) **`egrep -nf wz test.txt`**
- wystąpienia słów Karol i Karolina, zrób to na dwa sposoby **`egrep '(Karol|Karolina)' test.txt`** lub **`egrep 'Karol(ina)*' test.txt`**
- wystąpienia wieku czyli np.: 17 lat **`egrep '[0-9]+ lat. ' test.txt`**
- wiersze z jednym z podanych słów: *Mama, Karolina* **`fgrep 'Mama Karolina' test.txt`**

Polecenie cut

Składnia polecenia cut występuje w dwóch wariantach. Pierwszy służy do wybierania listy kolumn:

`cut –clista plik`

Polecenie w tej postaci kopiuje na standardowe wyjście pionowe kolumny inaczej znaki na pozycjach podanych na liście. Listę kolumn podajemy jako zakres 1-2 lub wyliczając 1,2. Spacje i tabulacje traktowane sa jak wszystkie inne znaki i zajmują po jednym bajcie.

Drugi przedstawia się następująco:

`cut –fpola –dznak -s plik`

Na standardowe wyjście kopiowane są pola oddzielone w danych wejściowych separatorami. Dane są pobierane z pliku lub standartowego wejścia.

`-fpola` – lista pól, które zostaną skopiowane na standardowe wyjście, domyślnie separatorami pól są białe znaki (spacja i tabulator)

`-dznak` – używa znaku *znak* jako separatora pól

`-s` – omija linie niezawierające separatora.

Ćwiczenia

3. Wyświetl kolumny od 1-4 z pliku *test.txt*.
4. Wyświetl z pliku *test.txt* tylko pole 2, które jest oddzielone znakiem , a następnie pole 2 oddzielone znakiem i.

Polecenie sort

Polecenie to służy do sortowania i łączenia plików tekstowych. Pobiera linie z plików podanych jako parametry lub domyślnie ze standardowego wejścia, sortuje je i domyślnie wypisuje na standardowe wyjście. Domyślna kolejność sortowania jest zgodna a kodami ASCII i można ją zmieniać.

`sort –opcje plik`

`-c` – sprawdza czy plik wejściowy jest posortowany i wpisuje odpowiedni komunikat

`-m` – łączy pliki w podanej kolejności przy założeniu, że są one odpowiednio posortowane (wtedy działa znacznie szybciej niż przy sortowaniu i łączeniu)

`-u` – usuwa linie o powtarzających się kluczach (domyślnie kluczem jest cała linia)

`-o plik` – kieruje wyjście do pliku

`-d` – porównuje używając tylko liter cyfr i białych znaków

`-f` – wyłącza rozróżnienie na wielkie i małe litery

`-i` – ignoruje znaki niedrukowalne (spoza zakresu 040 – 0176 tabeli ASCII)

`-n` – porównanie numeryczne (według wartości arytmetycznej)

`-r` – odwrócona kolejność sortowania (malejąca)

`-tc` – znak *c* jest używany jako separator pól

`-b` – ignoruje białe znaki poprzedzające pola

Ćwiczenia

5. Sprawdź czy plik *test.txt* jest posortowany wg kodu ASCII.
6. Posortuj plik *test.txt* wg liter.
7. Posortuj plik *test.txt* wg liter malejąco i zapisz w pliku *test2.txt*.

Polecenie uniq

Polecenie **`uniq`** pobiera linie z wejścia i porównuje kolejne linie sąsiadujące ze sobą. Jeśli co najmniej dwie kolejne linie są identyczne, to usuwane są wszystkie oprócz jednej. Po takim przetworzeniu tekst jest kierowany na wyjście. Jeśli w linii polecenia podano dwie

nazwy plików to pierwszy jest wejściem a drugi wyjściem. Jeśli jest tylko jedna nazwa pliku to traktowane jest to jako wejście a wyjście jest standardowe, jeśli nie ma żadnej nazwy to wejście i wyjście jest standardowe.

uniq –opcje wejście wyjście

-c – wypisuje nie tylko linię, ale też ile razy została powtórzona
-d – wypisuje tylko linie powtarzające się
-u – wypisuje tylko linie unikatowe
-n – ignorowanie n pierwszych pól w linii
+n – ignorowanie n pierwszych znaków pola

Ćwiczenia

8. Utwórz plik *wykonawcy* zawierający następującą treść:

Abba
Abba
Pink Floyd
Pink Floyd
Queen
Queen
Queen
The Animals

9. Użyj polecenia **uniq** następująco:
   - **uniq *wykonawcy***
   - **uniq –u *wykonawcy***
   - **uniq –d *wykonawcy***
   - **uniq –c *wykonawcy***

   Zaobserwuj różnicę.
10. Utwórz plik *wers.txt* w katalogu zadanie zawierający następującą treść:

    Linia 111
    Linia 111
    Linia 111
    Wiersz 221
    Wiersz 221
    wers 333
    linia 111
    linia 111

    Następnie wyświetl linie i numery tych linii (za pomocą polecenia **grep**), które zawierają słowo *Linia*, będąc w katalogu ze swoim imieniem.
11. W pliku *ks_tel.txt* Pomiędzy *Grzegorzami* wpisz *Jozue Wal 78-2222988*, a pod *Romanem - Adas Rol 78-6378383*, oraz zmień wielkie litery na małe w: *Admin*, *Serwis*, *Biuro* (polecenie r w vi), w tych trzech liniach usuń też numer kierunkowy i myślnik. Następnie wypisać te linie z pliku zawierające imiona Jola i Jozue stosując polecenie **grep**.
12. Napisz polecenie, które z pliku *ks_tel.txt* wypisze linie zawierające imiona *Adas*, *Artur*.
13. Stosując filtr **grep** i symbol dopełnienia zbioru, napisz polecenie, które wypisze tylko wiersz: *Artur Artek 65-3390258*.
14. Wyświetl linie z pliku *ks_tel.txt* zawierające telefony zaczynające się od *78*, ale tylko użytkowników z imieniem na *J*.
15. Użyj konstrukcji \{m\} w celu wyświetlenia linii, które zawierają cztery dwójki występujące po sobie w pliku *ks_tel.txt*.
16. Używając polecenia **egrep**, spowoduj wyświetlenie na ekranie linii zawierających imiona *Jolanta* i *Teresa* z pliku *ks_tel.txt*. Polecenie powinno dopuszczać imiona pisane małą i wielką literą. Użyj wyrażenia regularnego z symbolem |, a następnie ze znakiem nowej linii.
17. Używając polecenia **fgrep**, wyświetl na ekranie wszystkie linie z pliku *ks_tel.txt* nie zawierające imion *Roman*, *Jolanta*, *Pawel*. Użyj opcji **–v**.

18. Używając polecenia **fgrep**, wyświetl na ekranie wszystkie linie z pliku *ks_tel.txt* zawierające znak dolara.
19. Używając polecenia **egrep**, wyświetl na ekranie wszystkie linie z pliku *ks_tel.txt* rozpoczynające się od wielkiej i małej litery. Zastosuj wyrażenie regularne ze znakiem **+**.
20. Używając polecenia **egrep**, wyświetl na ekranie wszystkie linie z pliku *ks_tel.txt* zawierające numer telefonu. Następnie wypisz te linie w których numer telefonu zawiera znak -. Zastosuj wyrażenie regularne ze znakiem **+** i **?**.
21. Używając polecenia **grep**, wyświetl na ekranie wszystkie linie z pliku *ks_tel.txt* zawierające numer telefonu 4055522. Zastosuj podwyrażenia.
22. Używając polecenia **egrep**, wyświetl na ekranie wszystkie linie z pliku *ks_tel.txt* zawierające numer telefonu z dwoma piątkami. Zastosuj wyrażenie regularne z ().
23. Za pomocą polecenia **cut**, wyświetl kolumny od 1-10 z pliku *ks_tel.txt*.
24. Za pomocą polecenia **cut**, wyświetl pole 1i 2 z pliku *ks_tel.txt*. Separatorem jest spacja.
25. Jak zmodyfikować polecenie z 24, aby nie wyświetlał linii nie zawierających separatora.
26. Używając polecenia **sort**, sprawdź, czy plik *ks_tel.txt* jest posortowany wg kodów ASCII.
27. Używając polecenia **sort**, posortuj plik *ks_tel.txt* wg pól zawierających imiona.
28. Zrób to samo co w 27 tylko wyniki zapisz do pliku *sort.txt*. Jakim poleceniem sprawdzić zawartość nowego pliku?
29. Używając polecenia **uniq**, wypisz z pliku *wers.txt* kolejno:
    - wszystkie niepowtarzające się linie
    - wszystkie linie z liczbą wystąpień tej linii w pliku
    - tylko powtarzające się linie

Polecenie tr

Polecenie **tr** służy do usuwania lub zastępowania znaków. Kopiuje znaki ze standardowego wejścia na standardowe wyjście, zastępując po drodze niektóre z nich lub je usuwając.

tr –opcje łańcuch1 łańcuch2

-c – uzupełnienie znaków niewystępujących w *łańcuchu1* znakami z *łańcucha2*. Ze znaków niewystępujących w łańcuchu1 tworzony jest zestaw znaków ze zbioru o kodach **ASCII 0-255**.

-d – kasuje z tekstu wejściowego znaki podane w *łancuchu1*.

-s – jeżeli znak zawarty w *łańcuchu2* wystąpi w tekście wyjściowym kilka razy pod rząd, to jego wielokrotność jest usuwana (zostaje wpisany tylko jeden znak).

Przykłady

9. Polecenie **tr abc xyz** spowoduje zastąpienie wszystkich liter *a* przez *x*, *b* przez *y*, *c* przez *z*. Jeśli drugi łańcuch zawiera mniej liter niż pierwszy to zastępowane są tylko początkowe znaki, a kolejne nie ulegają zmianie. Jeśli znak występuje więcej niż raz w pierwszym łańcuchu np. **tr ala pas** wykonane będzie ostatnie podane zastąpienie, czyli *a* zostanie podmienione przez *s* a nie przez *p*. Aby zastąpić wszystkie małe litery wielkimi wystarczy wpisać **tr "[a-z]" "[A-Z]"** oba polecenia są w cudzysłowach ponieważ są to dwa różne argumenty polecenia **tr**. Możemy również użyć zapisu **tr abcde "[$*5]"** co oznacza zastąp pięć podanych znaków przez symbol *$*.

Edytor strumieniowy sed

Edytor **sed** kopiuje podane pliki (lub domyślnie standardowe wejście) na standardowe wyjście, podczas tej pracy, linia po linii, edytuje przenoszone dane wg odpowiednich komend. Polecenia takie mogą być zapisane w pliku, tworząc skrypt, mogą być zapisane w wierszu poleceń (można obie wersje łączyć).

sed –opcje 'polecenie' plik

sed –opcje –f skrypt plik

-n - wyłącza wyświetlanie przetwarzanych linii

-e polecenie – podaje polecenie/polecenia edycyjne, każde musi być poprzedzone opcję **–e**

-f - wczytuje polecenia edycyjne z pliku, jeśli podamy kilka plików to polecenia edycyjne zostaną wykonane w kolejności podania plików

Edytor wczytuje do wewnętrznego bufora po kolei linie z pliku wejściowego, następnie sprawdza, czy polecenie odnosi się do danej linii. Jeśli tak, to wykonuje polecenie na niej, następnie sprawdza kolejne polecenie, jeśli odnosi się do tej linii znowu je wykona, trzeba pamiętać, że linia ta jest wciąż modyfikowana i następne polecenia pracują już ze zmodyfikowaną linią, a nie tą wczytaną. Gdy wszystkie polecenia zostaną wykonane, zawartość bufora jest kopiowana na standardowe wyjście.
Każde polecenie edycyjne składa się z adresu i instrukcji. Kiedy adres nie zostanie podany polecenie zostanie wykonane dla wszystkich linii. Jako adres można podać numer linii (linie liczone są w sposób ciągły, dla wszystkich plików wejściowych), lub symbol **$** (ostatni wiersz), lub użycie wyrażenia regularnego. Podając dwa adresy oddzielone przecinkiem, określamy zakres linii, dla których zostaną przeprowadzone zmiany. Jeżeli drugi adres będzie wyrażeniem regularnym, jego poprawność zostanie sprawdzona od linii następującej po linii, określonej przez pierwszy adres. Jeżeli drugi adres zawiera numer równy lub mniejszy niż sprawdzana linia, edycji ulegnie tylko jedna linia. Jeśli po adresie wystąpi znak **!**, oznacza to, że polecenia będą wykonywane na wierszach, które nie są zgodne z adresem. Należy zawsze pamiętać, że adres będzie sprawdzany w buforze, który mógł już ulec modyfikacji poprzednim poleceniem edycyjnym.

Podstawowe polecenia edytora sed

Polecenie	**liczba adresów**	**Opis**
#	0	Komentarz.
:	0	Etykieta.
=	1	Wypisuje na standardowym wyjściu numer aktualnej linii.
a\tekst	1	Dodaje podany tekst na wyjście przed odczytaniem następnej linii.
d	2	Kasuje bufor i zaczyna następny cykl.
i\tekst		Wstawia tekst przed każdym tekstem zgodnym z adresem.
n	2	Wysyła bufor na standardowe wyjście i wczytuje nową linię do bufora.
P	2	Kopiuje zawartość bufora na standardowe wyjście (najczęściej używa się tego w połączeniu z opcją **sed –n**).
Q	1	Wyjście powoduje ominięcie następnych poleceń oraz zakończenie działania **sed**.
r plik	2	Wczytuje *plik* i wysyła jego zawartość na standardowe wyjście, po czym wczytuje następną linię.
s/wyrażenie/↵tekst/flaga	2	Zmienia *wyrażenie* na *tekst*. Liczba zmian zależy od podanej flagi: **n** – zmiana *n* kolejnych wystąpień wyrażenia (n=1..512) **g** – zmiana wszystkich wystąpień wyrażenia **p** – powoduje wyświetlanie bufora, jeżeli dokonano zmian w pliku, albo powoduje zapisanie bufora do pliku, jeżeli dokonano w nim zmian.
t etykieta	2	Powoduje skok do miejsca oznaczonego etykietą, pod warunkiem, że od początku działania skryptu lub wcześniejszego skoku wykonano jakieś zmiany tekstu w buforze. Jeśli nie podano żadnej etykiety, skrypt zostaje zakończony i zaczyna się następny cykl.
y/łańcuch1/↵łańcuch2	2	Wymienia znaki z *łańcucha1* na znaki z *łańcucha2*. Liczba znaków w obu łańcuchach musi być taka sama.
{}	2	Nawiasy te służą do grupowania poleceń w obrębie tego

samego adresu. Każde polecenie ma być oddzielną linią. Znak } też ma być w oddzielnej linii.

Jeśli nie chcemy aby powłoka interpretowała znaki typu ***** lub **[**, musimy w wierszu poleceń, przy wywołaniu edytora **sed**, zapisać całe polecenie w apostrofach.

Składnia polecenia edycyjnego:

```
adres,adres! instrukcja argumenty
```

Przykłady

10. Aby wyświetlić 15 pierwszych linii z pliku tekst.txt używamy polecenia: **`sed 15q tekst.txt`**, jeśli chcemy wyświetlić tylko ostatnią linię: **`sed '$!d' tekst.txt`**, aby wyświetlić linie o numerach 2-5 i 7-10 wpisujemy: **`sed –n –e '2,5p' –e '7,10p' tekst.txt`**.

Ćwiczenia

30. Korzystając z edytora *sed*, wyświetl pięć pierwszych linii pliku *ks_tel.txt*.
31. Za pomocą edytora *sed*, wyświetl zawartość pliku *ks_tel.txt* pomijając linie od 3 do 16.
32. Za pomocą edytora *sed*, wyświetl tylko ostatnią linie pliku *ks_tel.txt*.
33. Za pomocą edytora *sed*, wyświetl linie zawierające imiona rozpoczynające się na J w pliku *ks_tel.txt*.
34. Za pomocą edytora *sed*, wyświetl zawartość pliku *ks_tel.txt*, ale tak aby pierwsze trzy linie powtarzały się.
35. Zmodyfikuj polecenie z ćwiczenia 34 tak aby pokazały się tylko trzy pierwsze linie.

Filtr tekstowy awk

Program **`awk`** jest zaawansowanym filtrem tekstowym wyposażonym w specjalny język programowania zawierający zamienne, pętle i instrukcje warunkowe. Zdaniem programu jest przeglądanie plików i wyszukiwanie w nich zadanego wzorca, na którym zostaną wykonane zadane przez nas operacje. Wywołanie programu można zrealizować następująco:

```
awk –opcje program(parametry) plik
awk –opcje –f plik_programu(parametry) plik
```

`-Fz` – znak z staje się separatorem pól

`-f plik_programu` – ciąg instrukcji zostanie pobrany z pliku

`-v zmienna=wartość` – przypisanie zmiennej **`zmienna`** wartości **`wartość`** przed wykonaniem programu

Każda linia tekstu jest porównywana z wzorcem i jeżeli wzorzec w linii wystąpi, wykonana jest na nim operacja. Jeśli nie podamy wzorca, operacja zostanie przeprowadzona dla wszystkich linii. Jeśli nie określimy operacji to linie pasujące do wzorca zostaną przesłane na ekran. Jeśli polecenia zostaną zapisane w wierszu poleceń jako program trzeba ująć je w apostrof.

Ważniejsze operacje programu

Akcja	Opis
`exit(kod)`	Zatrzymanie wykonywania programu i zwrócenie kodu wyjścia **`kod`**.
`length` **`↦ (napisy)`**	Obliczanie długości tekstu lub zmiennej podanej jako argument. Jeżeli nie podano argumentu, domyślnie przyjmowana jest zmienna **`$0`**.
`print`	Wyświetlanie tekstu lub wartości zmiennej. Jeżeli niepodano żadnego argumentu, wyświetlana jest cała bieżąca linia (zmienna **`$0`**).
printf wzór wyraz	Działa podobnie jak **`print`**, wyświetlając podane po nim ciągi znaków (**`wyraz`**), ale można dodatkowo zdefiniować format (**`wzór`**) wyświetlania wyrażeń: **`%d`** – dziesiętna liczba całkowita **`%f`** – liczba zmiennoprzecinkowa (ang. *float*) **`%e`** – liczba w postaci wykładniczej **`%x`** – liczba w postaci szesnastkowej **`%c`** – jeden znak **`%s`** – ciąg znaków

substr ↱(a,b,c)	Zwraca ciąg **c** znaków z łańcucha **a**, zaczynając od znaku nr **b** (**b** i **c** są liczbami całkowitymi)
sprintf ↱(wzór, wyraz)	Działa podobnie jak **printf**, ale utworzony tekst można przypisać do zmiennej
system ↱(polecenie)	Powoduje wykonanie polecenia przez powłokę

Polecenie **awk** rozróżnia trzy typy wzorców: wyrażenia regularne, wyrażenia z użyciem operatorów relacyjnych i wzorce specjalne. Dwa pierwsze typy wzorców można z sobą łączyć za pomocą operatorów logicznych.
Wyrażenia relacyjne służą do porównywania liczb lub ciągów znaków. Używane są do tego następujące operatory:
== - równy
!= - różny
< - mniejszy
<= - mniejszy lub równy
> - większy
>= - większy lub równy
~ - dopasowanie wyrażenia regularnego
!~ - niedopasowanie (różnica) wyrażenia regularnego
Wzorce specjalne to **BEGIN** i **END** służą do określenia operacji, które zostaną wykonane przed przetworzeniem pierwszej linii i po przetworzeniu ostatniej linii danych wejściowych.
Operatory logiczne, pozwalające łączyć wyrażenia regularne i relacyjne, to:
! - negacja
&& - iloczyn logiczny (**AND**)
|| - suma logiczna (**OR**)
wyrażenia można grupować w nawiasy.
Program **awk** umożliwia, tak języki programowania, korzystanie ze zmiennych. W tym wypadku nie trzeba ich deklarować, gdyż tworzą się w momencie pierwszego użycia. Istnieją także zmienne specjalne, najczęściej używane to:
$1, **$2**, itd. - określają pola w aktualnie przetwarzanej linii
$0 - określa całą linie.
Numery zmiennych nie muszą być stałą i dozwolony jest zapis:
n=5 print $n - wyświetli to piąte pole w linii.
Inne zmienne to:
NF – liczba pól w aktualnej linii
NR – liczba linii w pliku
FS – separator pól
OFS – separator pól w pliku wyjściowym
ORS – separator rekordów w pliku wyjściowym.
Oprócz zmiennych można tworzyć tablice jednowymiarowe.

Ćwiczenia

36. Korzystając z edytora *awk*, wypisz z pliku *ks_tel.txt* osoby, których numery telefonów zaczynają się od 22. Popraw w pliku *Teresa Panek* na *Teresa Kanek*.
37. Korzystając z edytora *awk*, wypisz z pliku *ks_tel.txt* osoby, których numery telefonów zaczynają się od 22, a ich nazwiska zaczynają się na literę K.
38. Korzystając z edytora *awk*, wypisz całe linie z pliku *ks_tel.txt*, w których pierwsze pole zawiera litery, a drugie ich nie zawiera.

Tradycyjne narzędzie unixowe używane do połączeń pomiędzy terminalami przez sieć to **telnet**. Program ten łączy się z **demonem telnet** na zdalnym komputerze, wymienia z nim dane identyfikacyjne a następnie tworzy połączenie. Połączenie to jest jawne co oznacza, że wszystkie informacje przesyłane tym protokołem pomiędzy klientem a serwerem są dostępne dla każdego, kto ma dostęp do sieci. W Ubuntu jest dostępny klient programu **telnet** (instalowany domyślnie w systemie) do połączeń ze starszymi systemami. Nowszą, nowocześniejszą i bezpieczniejszą alternatywą telnetu jest program **ssh**. Program ten jest również instalowany domyślnie w Ubuntu wraz z innymi aplikacjami korzystającymi z tego protokołu. Za pomocą **ssh** można ustanowić zdalna sesję z dowolnym zdalnym komputerem, na którym działa **demon ssh**.

ssh adres
ssh użytkownik@adres

Podczas pierwszego połączenia zostanie wyświetlony komunikat z zapytaniem czy chcemy, rzeczywiście się połączyć. Należy odpowiedzieć ***yes*** i potwierdzić odpowiedź **enterem**. Wtedy narzędzie **ssh** doda klucz **RSA** do listy znanych hostów znajdujących się na tym komputerze i wyświetli komunikat o podanie hasła. Po podaniu poprawnego hasła, system wyświetli komunikat potwierdzający zalogowanie w systemie.
Wyjście z systemu to **logout**.
Będąc zalogowanym można zmienić hasło użytkownika:
passwd - zmienia hasło loginu na którym jesteś zalogowany

Możemy tez używać bezpiecznego narzędzia **scp** umożliwiającego transfer plików z jednego komputera na inny za pośrednictwem bezpiecznego, szyfrowego połączenia **ssh** i **sftp**. Zasada jest taka sama jak z **cp** tylko kopiowanie odbywa się z jednego serwera na drugi.

scp –opcja użytkownik@adres:plik_zrodlowy plik_docelowy
scp –opcja plik_zrodlowy uzytkownik@adres:plik_docelowy

Przykłady

1. wysłanie pliku `tekst.txt`
   scp tekst.txt ola@158.75.5.136:. lub
   scp tekst.txt ola@fuw.edu.pl:~/ola.
2. pobranie pliku `tekst.txt`
   scp ola@158.75.5.136:tekst.txt .
3. pobranie całej zawartości (rekurencyjnie) z katalogu `gry` do bieżącego katalogu
   scp -r ola@fuw.edu.pl:/gry .

Ćwiczenia

1. Stwórz w katalogu zadanie katalog *kop* a w tym katalogu plik *test* następnie
   - będąc w tym katalogu skopiuj ten plik do swojego katalogu domowego na serwerze uniwersyteckim (użytkownik@tempac.fuw.edu.pl)
   - otwórz nowy terminal i zaloguj się na swoje konto na serwerze uniwersyteckim, sprawdź czy plik test znajduje się w Twoim katalogu domowym
   - będąc na swoim koncie zmień nazwę tego pliku na *testpow*
   - będąc w katalogu *kop* na komputerze w pracowni skopiuj plik *testpow* na komputer
   - będąc na swoim koncie zmień hasło
   - będąc na swoim koncie zmień skopiuj cały katalog *kop* do siebie
   - wyloguj się z konta uniwersyteckiego