Python w Elektronicznej Sieci #5: Więcej na temat Unix'a

Projekt „Matematyka dla Ciekawych Świata",
Robert Ryszard Paciorek
<rrp@opcode.eu.org>

2020-03-24

1 Operacje na plikach

1.1 grep i wyrażenia regularne

Polecenie `grep [opcje] wyrażenie [plik1 [plik2 [...]]]` wyszukuje pasujące do wyrażenia regularnego wyrażenie linie w plikach, przydatne opcje:
`-v` zamiast pasujących wypisz nie pasujące
`-i` ignoruj wielkość liter
`-a` przetwarzaj pliki binarne jak tekstowe
`-E` korzystaj z „*Extended Regular Expressions*" (ERE) zamiast „*Basic Regular Expressions*" (BRE)

`-P` korzystaj z „*Perl-compatible Regular Expressions*" (PCRE) zamiast „*Basic Regular Expressions*" (BRE)

`-r` rekursywnie przetwarzaj podane katalogi wyszukując w wszystkich znalezionych plikach
`-R` jak -r, ale zawsze podąża za linkami symbolicznymi
`--exclude="wyrażenie"` pomiń pliki pasujące do wyrażenie (może zawierać znaki uogólniające powłoki)

`-l` wypisuje pliki z pasującymi liniami
`-L` wypisuje pliki z bez pasujących linii

Wyrażenia regularne[1] konstruuje się w oparciu o następujące znaki specjalne:

```
.      - dowolny znak
[a-z]  - znak z zakresu
[^a-z] - znak z poza zakresu (aby mieć zakres z ^ należy dać go nie na początku)
^      - początek napisu/linii
$      - koniec napisu/linii

*      - dowolna ilość powtórzeń
?      - 0 lub jedno powtórzenie
+      - jedno lub więcej powtórzeń
{n,m}  - od n do m powtórzeń

()     - pod-wyrażenie (może być używane dla powtórzeń, a także referencji
 ↪ wstecznych)
```

1.2 sed i inne narzędzia przetwarzania tekstów

- `sed [opcje] [pliki]` edytuje plik zgodnie z podanymi poleceniami, przydatne opcje:
  `-e "polecenie"` - wykonuj na pliku polecenie (może wystąpić wielokrotnie celem podania wielu poleceń)

1. Podana składnia dotyczy „*Extended Regular Expressions*", przy BRE niektóre z znaków sterujących wymagają zabezpieczenia odwrotnym ukośnikiem.

`-f "plik"` - wczytaj polecenia z pliku plik
`-E` - używaj rozszerzonych wyrażeń regularnych
`-i` - modyfikuj podany plik zamiast wypisywać zmieniony na stdout
przydatne polecenia:
`s@regexp@napis@[g]` - wyszukaj dopasowania do wyrażenia regularnego regexp i zastąp je przez napis, podanie opcji g powoduje zastępowanie wszystkich wystąpień a nie tylko pierwszego, znak `@` pełni rolę separatora i może zostać zamiast niego użyty inny znak
`y@zbiór1@zbiór2@` - zastąp znaki z zbiór1 znakami odpowiadającymi im pod względem kolejności znakami z zbiór2, znak `@` pełni rolę separatora i może zostać zamiast niego użyty inny znak
możliwe jest też m.in. adresowanie linii na których ma być wykonywana operacja, np: `0,/regexp/s@regexp@napis@` wykona polecenie s na liniach od początku pliku do linii pasującej do wyrażenia regularnego regexp, czyli zastąpi tylko pierwsze wystąpienie w pliku

- `tail [opcje] [plik]` wyświetla ostatnie linie pliku, przydatne opcje:
  `-n x` określa że ma zostać wyświetlone x ostatnich linii
  `-f` uruchamia dopisywania (gdy do pliku zostaną dopisane nowe linie tail je wyświetli)
- `head [opcje] [plik]` wyświetla początkowe linie pliku, przydatne opcje:
  `-n x` określa że ma zostać wyświetlone x pierwszych linii
- `diff ścieżka1 ścieżka2` porównuje pliki lub katalogi (w przypadku tych drugich porównuje ze sobą pliki o takich samych nazwach oraz zgłasza fakt występowania pliku tylko w jednym z katalogów), przydatne opcje:
  `-r` rekursywnie przetwarzaj podane katalogi
  `-u` wypisuje różnice w formacie "unified"
  `-c` wypisuje różnice w formacie "context"
- `patch` stosuje plik łaty (wynik diff'a) w celu zmodyfikowania plików, typowo:
  `patch -pn < plik.diff` co powoduje zastosowanie zmian opisanych w plik.diff na plikach w bieżącym katalogu, n określa ilość poziomów ścieżek podanych w pliku łaty które mają zostać zignorowane
- `sort [plik]` sortuje linie w wskazanym pliku, przydatne opcje:
  `-n` traktuj liczby jako wartości numeryczne a nie napisy
  `-i` ignoruj wielkość liter
  `-r` odwróć kolejność sortowania
  `-k n` sortuj wg kolumny n
  `-t sep` kolumny rozdzielane są przy pomocy separatora sep
- `cut [opcje] [pliki]` wybiera z pliku zadany zestaw kolumn, przydatne opcje:
  `-f nnn` wypierz kolumny określone przez nnn (np. 1,3-4,6- oznacza kolumnę 1, kolumny od 3 do 4 i od 6, a -3 oznacza 3 pierwsze kolumny)
  `-d sep` kolumny rozdzielane są przy pomocy separatora sep (musi być pojedynczym jedno bajtowym znakiem, aby ominąć to ograniczenie należy skorzystać z awk)
- `paste` łączy (odpowiadające sobie pod względem numerów) linie z dwóch plików
- `join` łączy linie z dwóch plików w oparciu o porównanie wskazanego pola
- `comm` porównuje dwa posortowane pliki pod względem unikalności linii (może wypisać wspólne lub występujące tylko w jednym z plików)
- `uniq` usuwa powtarzające się linie z posortowanego pliku, przydatne opcje:
  `-c` wypisz liczbę powtórzeń
  `-d` wypisz tylko linie z 2 lub więcej wystąpieniami
  `-u` wypisz tylko linie z 1 wystąpieniem

2 Przetwarzanie napisów

2.1 grep, cut, sed, ...

Jako że większość operacji wykonywanych w powłoce takiej jak bash wiąże się z uruchamianiem zewnętrznych programów, to także przetwarzanie napisów może być realizowane w ten sposób. Opiera się na tym jedno z podejść do obsługi napisów w bashu, którym jest korzystanie z standardowych komend POSIX, takich jak grep, cut, sed.

```
# obliczanie długości napisu w znakach, w bajtach i ilości słów w napisie
echo -n "ąbcć 123" | wc -m
echo -n "ąbcć 123" | wc -c
echo -n "ąbcć 123" | wc -w

# obliczanie ilości linii (dokładniej ilości znaków nowej linii)
wc -l < /etc/passwd

# wypisanie 5 pola (rozdzielanego :) z pliku /etc/passwd  z eliminacją
# pustych linii oraz linii złożonych tylko ze spacji i przecinków
cut -f5 -d: /etc/passwd | grep -v '^[ ,]*$'
# komenda cut wybiera wskazane pola, opcja -d określa separator
```

Inną bardzo przydatną komendą jest sed pozwala ona m.in na zastępowanie wyszukiwanego na podstawie wyrażenia regularnego tekstu innym:

```
echo "aa bb cc bb dd bb ee" | sed -e 's@\([bc]\+\) \([bc]\+\)@X-\2-X@g'
```

Sedowe polecenie s przyjmuje 3 argumenty (oddzielane mogą być dowolnym znakiem który wystąpi za s), pierwszy to wyszukiwane wyrażenie, drugi tekst którym ma zostać zastąpione, a trzeci gdy jest g to powoduje zastępowanie wszystkich wystąpień a nie tylko pierwszego.

Należy zwrócić uwagę na różnicę w składni wyrażenia regularnego polegającą na poprzedzaniu (,) i + odwrotnym ukośnikiem aby miały znaczenie specjalne (jeżeli nie chcemy tego robić możemy włączyć obsługę ERE w sed poprzez opcję -E).

Innymi przydatnymi komendami przetwarzającymi (specyficznej postaci) napisy są polecaenia basename i dirname. Służą one do uzyskania nazwy najgłębszego elementu ścieżki oraz ścieżki bez tego najglębszego elementu. Zobacz wynik działania:

```
basename /proc/sys/net/core/
dirname /proc/sys/net/core/
```

2.2 awk

Awk jest interpreterem prostego skryptowego języka umożliwiający przetwarzanie tekstowych baz danych postaci linia=rekord, gdzie pola oddzielane ustalonym separatorem (można powiedzieć że łączy funkcjonalność komend takich jak grep, cut, sed z prostym językiem programowania).

Wyżej zaprezentowane wypisanie 5 pola (rozdzielanego :) z pliku /etc/passwd z eliminacją pustych linii oraz linii złożonych tylko ze spacji i przecinków, realizowane przy użyciu poleceń cut i grep może być zrealizowane za pomocą samego awk:

```
awk -F: '$5 !~ "^[ ,]*$" {print $5}' /etc/passwd
```

Awk daje duże możliwości przy przetwarzaniu tego typu tekstowych baz danych – możemy np. wypisywać wypisywać pierwsze pole w oparciu o warunki nałożone na inne:

```
awk -F: '$5 !~ "^[ ,]*$" && $3 >= 1000 {print $1}' /etc/passwd
```

Jak widać w powyższych przykładach do poszczególnych pól odwołujemy się poprzez `$n`, gdzie `n` jest numerem pola, `$0` oznacza cały rekord

Program dla każdego rekordu przetwarza kolejne instrukcje postaci `warunek { komendy }`, instrukcji takich może być wiele w programie (przetwarzane są kolejno), komenda `next` kończy przetwarzanie danego rekordu.

Separator pola ustawiamy opcją `-F` (lub zmienną `FS`), domyślnym separatorem pola jest dowolny ciąg spacji i tabulatorów (w odróżnieniu od cut separator może być wieloznakowym napisem lub wyrażeniem regularnym). Domyślnym separatorem rekordu jest znak nowej linii (można go zmienić zmienną RS).

Awk jest prostym językiem programowania obsługującym podstawowe pętle i instrukcje warunkowe oraz funkcje wyszukujące i modyfikujące napisy:

```
echo "aba aab bab baa bba bba" | awk '{
        # dla każdego pola w rekordzie
        for (i=1; i<=NF; ++i) {
                # jeżeli jego numer jest parzysty
                # to zastąp wszystkie ciągi b pojedynczym B
                if (i%2==0)
                        gsub("b+", "B", $i);

                # wyszukaj pozycję pod-napisu B
                ii = index($i, "B")
                # jeżeli znalazł
                # to wypisz pozycję i pod-napis od tej pozycji do końca
                if (ii)
                        printf("# %d %s\n", ii, substr($i, ii))
                # zwróć uwagę że w AWK liczy elementy napisy od 1 a nie od 0
        }
        print $0
}'
```

AWK obsługuje także tablice asocjacyjne pozwala to np. policzyć powtórzenia słów:

```
echo "aa bb aa ee dd aa dd" | awk '
        BEGIN {RS="[ \t\n]+"; FS=""}
        {slowa[$0]++}
        # może być kilka bloków {} pasujących do rekordu
        # jeżeli nie użyjemy next przetworzone zostaną wszystkie
        # {printf("rekord: %d done\n", NR)}
        END {for (s in slowa) printf("%s: %s\n", s, slowa[s])}
'
```

Podobny efekt możemy uzyskać stosując ”uniq -c” (który wypisuje unikalne wiersze wraz z ich ilością) na odpowiednio przygotowanym napisie (spacje zastąpione nową linią, a linie posortowane):

```
echo "aa bb aa ee dd aa dd" | tr ' ' '\n' | sort | uniq -c
```

Jednak rozwiązanie awk można łatwo zmodyfikować aby wypisywało pierwsze wystąpienie linii bez sortowania pliku.

3 Praca zdalna

3.1 powłoka zdalna

Komenda `ssh [user@]host` umożliwia uzyskanie powłoki zdalnego systemu poprzez szyfrowane połączenie, przydatne opcje:
`-L portLokalny:hostZdalny:portZdalny` tworzy tunel przekierowujący dane kierowane na portLokalny komputera na którym działa klient ssh do portu portZdalny na serwerze hostZdalny poprzez serwer SSH (przydatne gdy hostZdalny jest osiągalny z hostSSH ale nie z komputera lokalnego)
`-D port` tworzy tunel dynamiczny na wskazanym porcie (może on być użyty jako proxy typu SOCKS np. w Firefoxie w celu zapewnienia dostępu do zasobów WWW dostępnych z serwera SSH a niedostępny z komputera lokalnego)
`-p port` określa inny niż domyślny port serwera SSH
`-X` aktywuje przekazywanie komend X serwera ze strony zdalnej do klienta (pozwala na uruchomienie po stronie zdalnej aplikacji z GUI, które zostanie wyświetlone na lokalnym X serwerze)

3.2 zdalne kopiowanie

Najprostszą metą kopiowania plików pomiędzy różnymi systemami jest wykorzystanie do tego ssh, typowo robi się to na jeden z 3 sposobów:

- poleceniem `scp [opcje] źródło1 [źródło2 [...]] cel` , które kopiuje wskazany plik (lub pliki) do wskazanej lokalizacji, w przypadku kopiowania wielu plików cel powinien być katalogiem, do ważniejszych opcji należy zaliczyć:
  `-r` pozwala na (rekursywne) kopiowanie katalogów
  `-P port` określa port SSH
  W odróżnieniu od `cp` źródło lub cel w postaci `[user@]host:[ścieżka]` wskazują na zdalny system dostępny poprzez SSH.
- poleceniem `rsync [opcje] źródło cel` , które kopiuje (synchronizacjiuje) pliki i drzewa katalogów (zarówno lokalnie jak i zdalnie), do ważniejszych opcji należy zaliczyć:
  `-r` pozwala na (rekursywne) kopiowanie katalogów
  `-l` kopiuje linki symboliczne jako linki symboliczne (zamiast kopiowania zawartości pliku na który wskazują)
  `-t` zachowuje czas modyfikacji plików
  `-u` kopiuje tylko gdy plik źródłowy nowszy niż docelowy
  `-c` kopiuje tylko gdy plik źródłowy i docelowy mają inne sumy kontrolne
  `--delete` usuwa z docelowego drzewa katalogów elementy nie występujące w drzewie źródłowym

  `-e 'ssh'` pozwala na kopiowanie na/z zdalnych systemów za pośrednictwem ssh, źródło lub cel w postaci `[user@]host:[ścieżka]` wskazują na zdalny system `--partial --partial-dir=."-tmp-"` zachowuje skopiowane częściowo pliki w katalogu .-tmp- (pozwala na przerwanie i wznowienie transferu pliku)
  `--progress` pokazuje postęp kopiowania
  `--exclude="wzorzec"` pomija (w kopiowaniu i kasowaniu) pliki pasujące do wzorzec (wzorzec może zawierać znaki uogólniające powłoki) `-n` symuluje pracę (pokazuje co zostałoby skopiowane, ale nie kopiuje)
- złożonego polecenia opartego na przekierowaniu wyjścia jakiejś komendy do ssh, które uruchamia po zdalnej stronie proces odbierający te dane na swoim standardowym wejściu, np.:
  - `tar -czf - ścieżka1 [ścieżka2 [...]] | ssh [user@]host 'cat > plik.tgz'` archiwizuje wskazane pliki/katalogi bezpośrednio na zdalny system z użyciem tar i kompresji gzip do pliku `plik.tgz`
  - `tar -cf - ścieżka1 [ścieżka2 [...]] | ssh [user@]host 'tar -xf - -C cel'` kopiuje wskazane pliki/katalogi na zdalny system z użyciem tar do katalogu `cel`

4 Użytkownicy, uprawnienia i procesy

4.1 Uprawnienia do plików

Podstawowe unixowe uprawnienia do plików składają się z trzech członów: uprawnienia dla właściciela (u), grupy (g) i pozostałych użytkowników (o). W każdym z członów mogą być przyznane uprawnienia do czytania (r), pisania (w) i wykonywania (x); w odniesieniu do plików jest to intuicyjne (uprawnienie do wykonywania jest potrzebne do uruchomienia programów), natomiast w stosunku do katalogów wygląda to następująco: uprawnienia do czytania pozwalają na listowanie zawartości, do wykonania pozwalają na dostęp, do zawartości katalogu (wejścia do niego) do pisania na tworzenie nowych obiektów wewnątrz niego i zmienianie nazw istniejących.

Rozszerzeniem podstawowych uprawnień opisanych powyżej jest mechanizm Filesystem Access Control List (ACL, fACL).

Wszystkie poniższe komendy przyjmują opcję `-R` powodującą rekursywne wykonywanie zmian na drzewku katalogów/plików rozpoczynającym się w podanej ścieżce.

- `chown [opcje] właściciel ścieżka` zmiana właściciela pliku
- `chgrp [opcje] grupa ścieżka` zmiana grupy do której należy plik pliku
- `chmod [opcje] uprawnienia ścieżka` zmiana prawa dostępu do pliku(ów)
- `getfacl [opcje] [ścieżka]` dczyt uprawnień związanych z listami kontroli dostępu fACL
- `setfacl [opcje] [ścieżka]` ustawianie uprawnień związanych z listami kontroli dostępu fACL
- `chattr` zmienia atrybuty plików związanych z systemem plików (np. zabrania jakiejkolwiek modyfikacji pliku)

4.2 Użytkownicy

- `id [użytkownik]` informacja o użytkowniku (m.in. grupy do których należy)
- `whoami` informacja o aktualnym użytkowniku
- `w lub who` informacja o zalogowanych użytkownikach
- `passwd [użytkownik]` zmiana hasła
- `su [użytkownik]` przełącza użytkownika (aby przełączony użytkownik miał dostęp do "naszego" x serwera wcześniej wydajemy `xhost LOCAL:użytkownik`)
- `sudo` program pozwalający na wykonywanie uprzywilejowanych komend przez wyznaczonych użytkowników

4.3 Procesy i zasoby

- `ps [opcje]` wyświetla aktualnie działające procesy i informacje o nich
  np. kombinacja opcji `-Alf` powoduje wyświetlenie wszystkich procesów, w długim, pełnym formacie wypisywania
- `top` monitorowanie procesów obciążających CPU, pamięć, itd
- `iotop` monitorowanie procesów obciążających I/O
- `kill [opcje] pid` przesyła sygnał do procesów o podanych PID
- `killall [opcje] nazwa` przesyła sygnał do procesów o pasującej nazwie

4.4 Planowanie zadań

Typowo system zapewnia usługę uruchamiania zadań o zadanym czasie. Z usługi tej można skorzystać przy pomocy poleceń:

- `crontab` pozwala oglądać i edytować tablice zaplanowanych zadań cyklicznych (dla cron'a)
- `at` pozwala jednorazowo zaplanować zadanie

Pliki konfiguracyjne crona / obsługiwane crontab-em mają postać: `minuty godzina dzienMiesiaca miesiac dzienTygodnia polecenie`. Wpis oznacza że polecenie ma zostać wykonane jeżeli wszystkie warunki będą spełnione, jeżeli jakiś warunek nie jest nam potrzebny można użyć gwiazdki `*`, z kolei `*/n` oznacza wykonywanie jeżeli dana wartość jest podzielna przez n. Np.: `*/20 3 * * 1 ls` oznacza wykonanie komendy ls w każdy poniedziałek o godzinie 3:00 3:20 i 3:40

Standardowe wyjście, wyjście błędu oraz powiadomienie o niezerowym kodzie powrotu domyślnie są wysyłane na lokalny adres mailowy użytkownika będącego właścicielem danego contaba. Niekiedy dostępny jest także `anacron` pozwalający na mniej precyzyjne planowanie zadań.

4.5 Struktura katalogów

Systemy unix'owe posiadają drzewiasty system plików zaczynający się w katalogu głównym oznaczanym przez ukośnik (`/`), w którym zamontowany jest główny system plików (rootfs), inne systemy plików mogą być montowane w kolejnych katalogach. Do najistotniejszych katalogów należy zaliczyć:

- `/bin` zawierający pliki wykonywalne podstawowych programów
- `/sbin` zawierający pliki wykonywalne podstawowych programów administracyjnych
- `/lib` zawierający pliki podstawowych bibliotek
- `/usr` zawierający oprogramowanie dodatkowe (wewnętrznie ma podobną strukturę do głównego - tzn. katalogi `/usr/bin`, `/usr/sbin`, `/usr/lib`, itd)
- `/etc` zawierający konfiguracje ogólnosystemowe
- `/var` zawierający dane programów i usług (takie jak kolejka poczty, harmonogramy zadań, bazy danych)
- `/home` zawierający katalogi domowe użytkowników (często montowany z innego systemu plików, dlatego też root ma swój katalog domowy w `/root`, aby był dostępny nawet gdy takie montowanie nie doszło do skutku)
- `/tmp` zawierający pliki tymczasowe (typowo czyszczony przy starcie systemu); w Linuxie występuje też `/run` przeznaczony do trzymania danych tymczasowych działających usług takich jak numery pid, blokady, itp
- `/dev` zawierający pliki reprezentujące urządzenia; w Linuxie występuje też `/sys` zawierający informacje i ustawienia dotyczące m.in. urządzeń
- `/proc` zawierający informacje o działających procesach (w Linuxie także interfejs konfiguracyjny dla wielu parametrów jądra)

Pliki i katalogi których nazwa rozpoczyna się od kropki traktowane są jako pliki ukryte.

Z punktu widzenia programisty czy też użytkownika (prawie) wszystko jest plikiem, których istnieją różne rodzaje (zwykły plik, katalog, urządzenie znakowe, urządzenie blokowe, link symboliczny, kolejka FIFO, ...); pewnym wyjątkiem są urządzenia sieciowe (które nie mają reprezentacji w systemie plików (ale gniazda związane z nawiązanymi połączeniami obsługuje się zasadniczo tak jak pliki).

5 Zadania

Zadanie 5.0.1

Wyświetl plik `/etc/passwd` z zastąpionym `false` przez `FALSE`.

Zadanie 5.0.2

Wyświetlić same nazwy (bez ścieżki ścieżką) wszystkich plików i katalogów znajdujących się bezpośrednio w `/etc/` których druga litera to `a` natomiast trzecia to `p` lub `s`. Przedstaw rozwiązanie z użyciem i bez użycia komendy `cd`.

Zadanie 5.0.3

Napisz polecenie które wyszuka wszystkie wystąpienia napisu `nameserver` w plikach znajdujących się w katalogu `/etc` (wraz z jego podkatalogami).

Zadanie 5.0.4

Zmodyfikuj rozwiązanie zadania 5.0.3 tak aby wyłącznie wypisywało nazwy plików (mogą być ze ścieżką) zawierających wyszukiwany napis.

Zadanie 5.0.5

Wyświetl z /etc/passwd linie w których UID (3 pole) ma warość >= 1000.

Zadanie 5.0.6

Polecenie `last` wypisuje informację o ostatnich zalogowaniach w systemie. Napisz polecenie (wykorzystujące `last`), które wypisze informację jak często logowali się poszczególni użytkownicy.

Zadanie 5.0.7

Wylistuj zawartość /etc/ z wskazanej maszyny zdalnej.

Zadanie 5.0.8

Skopiuj plik /etc/hostname z wskazanej maszyny zdalnej do /tmp/hostname.

6 Praca domowa

6.1 Instrukcja wysyłania rozwiązań

Rozwiązania zadań domowych należy przesłać na adres `licealisci.pracownia@icm.edu.pl` wpisując jako temat wiadomości `g2.x PD5`, gdzie `x` to numer grupy, np. `g2.1 PD5` dla grupy nr. 1, itd. Zadania domowe są nie obowiązkowe, jednak zachęcamy do ich robienia i wysyłania rozwiązań (nawet niekompletnych).

Termin nadsyłania zdań domowych to 2020-03-31 godzina 16^{00}. Jeżeli wysłałeś rozwiązania w terminie, ale nie były one w 100% poprawne i dostałeś od sprawdzającego możliwość wysłania poprawki masz na to dodatkowe 4 dni.

Na ten adres można także nadsyłać ewentualne pytania do zadań (zarówno domowych jak i innych zamieszczonych w skrypcie), w tym wypadku także prosimy o umieszczenie w temacie wiadomości `g2.x`, gdzie `x` to numer grupy.

6.2 Zadania domowe

Zadanie 6.2.1 — 1 pkt

Wyświetl posortowaną numerycznie, według 3go pola zawartość pliku `/etc/passwd`.

Zadanie 6.2.2 — 1 pkt

Wyświetl z pliku `/etc/passwd` loginy użytkowników (pierwsze pole), którzy mają ustawioną powłokę (pole 7) na `/bin/false`

Zadanie 6.2.3 — 4 pkt (2 pkt za rozwiązanie)

Wyświetl 20tą linię z pliku `/etc/passwd`. Przedstaw dwa różne (używające innych komend) rozwiązania.