Python w Elektronicznej Sieci #1: Wprowadzenie do Unix'a

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

2020-02-25

1 Tytułem wstępu

Kurs *„Python w Elektronicznej Sieci"* jest intensywnym wprowadzeniem w najważniejsze zagadnienia związane z systemami typu Unix, programowaniem, sieciami komputerowymi oraz podstawami elektroniki, która stoi za działaniem komputerów, sieci komputerowych i dużej części współczesnego świata. W ramach kursu zapoznasz się:

- z kilkoma z pośród najistotniejszych języków programowania (w tym z Pythonem i C),
- pracą w systemach typu Unix/Linux oraz ich działaniem,
- działaniem, budową i wykorzystywaniem sieci komputerowych,
- podstawami elektroniki analogowej i cyfrowej oraz programowania mikrokontrolerów.

Kurs przeznaczony jest dla osób zainteresowanych tą tematyką i posiadających elementarną wiedzą związaną z tymi zagadnieniami (podstawy programistyczne, wiedzę z fizyki z zakresu elektryczności, itp). Zagadnienia na kursie w miarę możliwości omawiane będą od podstaw, jednak ze względu na intensywność kursu omówienie podstaw należy traktować raczej jedynie jako przypomnienie. Celem kursu jest ułatwienie dalszego zgłębiania tajników szeroko rozumianej informatyki i elektroniki poprzez przekazanie gruntownych podstaw oraz ich uporządkowanie i usystematyzowanie.

2 Praca w terminalu

Podstawowym sposobem wydawania poleceń w systemach typu Unix jest wpisywanie ich w terminalu. Terminal może pracować w trybie tekstowym lub może być uruchomiony (jako tzw. emulator terminala) w trybie graficznym.

2.1 Komendy

Komendy składają się z nazwy polecenia oraz opcji i argumentów. Nazwą polecenia może być nazwa funkcji wbudowanej, nazwa programu (znajdującego się w ścieżce wyszukiwania programów) lub pełna ścieżka do programu. Po nazwie polecenia mogą występować opcje i/lub argumenty. Są one oddzielane od nazwy polecenia i od siebie przy pomocy spacji[1]. Nie ma silnego rozróżnienia opcji od argumentów, typowo stosowaną konwencją jest rozpoczynanie opcji od pojedynczego myślnika (opcje krótkie - jednoliterowe) lub dwóch myślników (opcje długie). W przypadku stosowania tej konwencji po pojedynczym myślniku może występować kilka bezargumentowych opcji jednoliterowych. Typowo argumenty opcji oddzielane są od nich spacją (w przypadku opcji krótkich) lub znakiem równości (w przypadku opcji długich). Jeżeli któryś z składników komendy (np. argument) zawiera spacje należy je zabezpieczyć przy pomocy odwrotnego ukośnika lub ujęcia zawierającego je napisu w apostrofy lub cudzysłowia.

1. zasadniczo dowolnego ciągu białych znaków: spacji, tabulatorów, „escapowanych" nowych linii.

2.2 Powłoka

2.2.1 bash

Bash jest chyba najpopularniejszą powłoką (interpreterem poleceń). Jest zgodny ze składnią z sh, zapewnia m.in. obsługę zmiennych (zasadniczo napisowych) oraz omówionych w dalszej części skryptu znaków uogólniających. Umożliwia edycję linii poleceń, dopełnianie komend, ścieżek, itp przy pomocy tabulatora oraz zapewnia dostęp do historii linii poleceń (poruszanie się po historii strzałkami, wyszukiwanie przy pomocy Ctrl+R, polecenie `history`).

2.2.2 screen i tmux

`screen` i `tmux` są tzw. multiplexerami terminala - pozwala na uzyskanie wielu okien konsoli (także np. wyświetlanych jedno obok drugiego) na pojedynczym terminalu. Ponadto pozwalają na odłączanie i podłączanie sesji, co pozwala na łatwe pozostawienie działającego programu po wylogowaniu i powrót do niego później.

2.3 Uzyskiwanie pomocy

Informację na temat działania danej komendy oraz jej opcji można uzyskać w wbudowanym systemie pomocy przy pomocy poleceń `man` lub `info` / `pinfo`. Większość poleceń obsługuje także opcje `--help` lub `-h`, które wyświetlają informację na temat ich użycia.

Do zapamiętania

Zarówno w tekstach pomocy jak i w tym dokumencie stosowana jest konwencja polegająca na oznaczaniu opcjonalnych argumentów poprzez umieszczanie ich w nawiasach kwadartowych (jeżeli podajemy ten argument do komendy nie obejmujemy go już tymi nawiasami) oraz rozdzielaniu alternatywnych opcji przy pomocy `|`. Np. `a [b] c|d` oznacza iż polecenie `a` wymaga argumentu postaci `c` albo `d`, który może być poprzedzony argumentem `b`.

2.4 more i less

Jeżeli wynik jakiejś komendy nie mieści się na ekranie do jego obejrzenia możemy użyć poleceń `more` lub `less`. Są to programy umożliwiające przeglądanie tekstu ekran po ekranie. `less` posiada większe możliwości od more (w szczególności posiada możliwość przeglądanie dokumentu w tył)[2]. Programy te kończą się po wciśnięciu klawisza `q`. `less` umożliwia także wyszukiwanie – klawisz `/` pozwala na wprowadzenie szukanej frazy, a `n` na wyszukanie kolejnego wystąpienia. Programy te umożliwiają też wyświetlanie wskazanych jako argumenty plików.

2.5 Przekierowania

Typowo program posiada trzy strumienie danych: jednen wejściowy (stdin) i dwa wyjściowe (stdout i stderr). Standardowe wyjście możemy przekierować na standardowe wejście innego programu przy pomocy `|`, np:

```
ls --help | less
```

Konstrukcja ta przekieruje wynik komendy `ls` uruchomionej z opcją `--help` do komendy `less`.

Możemy także przekierować standardowe wyjście do pliku (przy pomocy `>` lub `>>`, gdy chcemy dopisywać do pliku) lub pobrać standardowe wejście z pliku (przy pomocy `<`). `2>` pozwala na przekierowanie standardowego wyjścia błędu do pliku, a `>&` i `|&` pozwala na przekierowanie obu strumieni odpowiednio do pliku lub standardowego wejścia innego polecenia.

2. wybrane przydatne opcje: `-X` nie czyści ekranu przy wychodzeniu z less'a (całość historii wyświetlania pliku pozostaje w historii terminala) `-F` automatycznie kończy gdy wyświetlany tekst mieści się na jednym ekranie

2.6 vi i vim

`vi` jest chyba najbardziej zaawansowanym edytorem, którego obecność gwarantuje standard POSIX. `vim` jest mocno rozbudowanym jego klonem, oferującym bardzo zaawansowane funkcjonalności, powszechnie stosowanym jako zamiennik oryginalnego `vi`. `vim` obsługuje 3 podstawowe tryby pracy: komend (służący do wydawania opisanych niżej poleceń), wizualny (służący do zaznaczania i wydawania niektórych komend), edycji (wstawiania/nadpisywania - służący do wprowadzania tekstu). Podstawowa klawiszologia:

- `Esc` powrót do trybu komend
- `i` tryb wstawiania
- `R` tryb zastępowania
- `Insert` zmiana trybu wstawiania i zastępowania
- `v` tryb wizualny (umożliwia zaznaczenie przy pomocy strzałek)
- `y` skopiuj; `d` - wytnij (skopiuj i usuń)
  po `y`, `d` można podać np. `20l` lub `20[strzałka w prawo]` co oznacza 20 kolejnych znaków, `2w` oznacza dwa słowa
- `x` wytnij (skopiuj i usuń) znak (może być poprzedzone ilością znaków do wycięcia)
- `yy` skopiuj linię; `dd` - wytnij (skopiuj i usuń)
  w obu wypadkach może być poprzedzone ilością linii do skopiowania/wycięcia
- `p` wkleja po; `P` - wkleja przed
- `u` cofa ostatnią operację
- `/` szukanie
- `n` wyszukanie następnego wystąpienie; `N` wyszukanie poprzedniego wystąpienie
- `G` przejście do wskazanej linii, numer podajemy przed G, 0 oznacza ostatnią linię w pliku, więc `0G` spowoduje przejście do niej
- `:[zakres]s@regexp@napis@[g]` wyszukaj i zastąp wyrażenie regularne regexp przez napis; zakres może być:
  - numerem linii,
  - przedziałem z numerami linii postaci `pierwsza,ostatnia`, gdzie:
    `.` oznacza bieżącą linię, `$` oznacza ostatnią linię w pliku, wartość numeryczna poprzedzona `+` oznacza tyle kolejnych linii od bieżącej, a poprzedzona `-` przed bieżącą,
  - znakiem `%` (co oznacza cały plik),
  - zakresem zaznaczonym w trybie wizualnym;

  podanie opcji g powoduje zastępowanie wszystkich wystąpień a nie tylko pierwszego;
  znak `@` pełni rolę separatora i może zostać zamiast niego użyty inny znak
- `:e ścieżka` otwarcie wskazanego pliku
- `:w` zapis (można także podać ścieżkę pod jaka ma zostać zapisany plik)
- `:q` wyjście ; `:q!` wyjście bez zapisywania ; `:wq` zapis i wyjście
- `:n` - następny plik ; `:N` - poprzedni plik
- `:split` - podział okna ; `:vs` - pionowy podział okna ; `Ctrl+W` a następnie strzałka - przełączanie między oknami
- `:%!xxd` pokazanie wartości numerycznych i umożliwienie edycji pliku jako binarnego;
  `:%!xxd -r` powrót do normalnej edycji

3 Operacje na systemie plików

3.1 Podstawowe komendy

3.1.1 katalog roboczy

- `cd [ścieżka]` zmiana bieżącego katalogu,
  warto zauważyć, iż katalogi w ścieżce oddzielamy ukośnikami `/`, bieżący katalog oznaczamy kropką `.`, nadrzędny oznaczamy dwiema kropkami `..`,
  ścieżki zaczynające się od ukośnika `/` oznaczają *ścieżki bezwzględne* (od korzenia systemu plików), pozostałe oznaczają *ścieżkę względną* (wyrażoną względem bierzącego katalogu),
  katalog domowy oznacza się tyldą `~`
- `pwd` wyświetla ścieżkę do bieżącego katalogu

3.1.2 wyświetlanie i wyszukiwanie

- `ls [opcje] [ścieżka]` listowanie zawartości katalogu, do ważniejszych opcji należy zaliczyć:
  `-a` wyświetlaj pliki ukryte (zaczynających się od kropki)
  `-l` wyświetlaj pliki w formie listy z szczegółowymi informacjami (uprawnienia, rozmiar, data modyfikacji, właściciel, grupa, rozmiar)
  `-1` wyświetlaj pliki w formie 1 plik w jednej linii (bez dodatkowych informacji; stosowane domyślne gdy wynik komendy przekazywany jest strumieniem do innej komendy lub pliku)
  `-h` stosuj jednostki typu k, M, G zamiast podawać rozmiar w bajtach
  `-t` sortuj wg daty modyfikacji
  `-S` sortuj wg rozmiaru
  `-r` odwróć kolejność sortowania
  `-c` użyj daty utworzenia zamiast daty modyfikacji (stosowane w połączeniu z `-l` i/lub `-t`)
  `-d` wyświetlaj informacje o katalogu zamiast jego zawartości
- `find [opcje] [katalog startowy] [wyrażenie]` wyszukiwanie w systemie plików w oparciu o nazwę/ścieżkę lub właściwości pliku, do ważniejszych opcji należy zaliczyć:
  `-P` wypisuj informacje o linkach symbolicznych a nie plikach przez nie wskazywanych (domyślne)
  `-L` wypisuj informacje o wskazywanych przez linki symboliczne plikach
  do ważniejszych elementów wyrażenia należy zaliczyć:
  `-name "wyrażenie"` pliki których nazwa pasuje do wyrażenia korzystającego z shellowych znaków uogólniających
  komenda find (w odróżnieniu np. od ls) samodzielnie interpretują wyrażenia zawierające shellowe znaki uogólniające, w związku z czym konieczne może się okazać zabezpieczenie ich przed interpretacją przez powłokę np. przy pomocy umieszczenia wewnątrz pojedynczych cudzysłowów
  `-iname "wyrażenie"` jak -name, tyle że nie rozróżnia wielkości liter
  `-path "wyrażenie"` pliki których ścieżka pasuje do wyrażenia korzystającego z shellowych znaków uogólniających
  `-ipath "wyrażenie"` jak -path, tyle że nie rozróżnia wielkości liter
  `-regex "wyrażenie"` pliki których ścieżka pasuje do wyrażenia regularnego
  `-iregex "wyrażenie"` jak -regexp, tyle że nie rozróżnia wielkości liter

  `warunek -o warunek` łączy warunki sumą logiczną OR zamiast domyślnego iloczynu logicznego AND
  `! warunek` negacja warunku

  `-mtime [+|-]n` pliki których modyfikacja odbyła się n*24 godziny temu
  `-mmin [+|-]n` pliki których modyfikacja odbyła się n minut temu
  `-ctime [+|-]n` pliki które zostały utworzone n*24 godziny temu
  `-cmin [+|-]n` pliki które zostały utworzone n minut temu
  `-size [+|-]n[c|k|M|G]` pliki których rozmiar wynosi n (c - bajtów, k - kilobajtów, M - Megabajtów, G - gigabajtów)

w powyższych testach + oznacza więcej niż, - oznacza mniej niż, uwaga: porównywaniu podlegają liczby całkowite, np. +1 oznaczacza > 1 ***w liczbach całkowitych tzn.*** $\geqslant 2$

`-exec polecenie \{\} \;` dla każdego znalezionego pliku wykonaj `polecenie` podstawiając ścieżkę do tego pliku pod `\{\}` (zastosowane odwrotne ukośniki służą zabezpieczeniu nawiasów klamrowych i średnika przed zinterpretowaniem ich przez powłokę)
`-execdir polecenie \{\} \;`, podobnie jak `-exec` tyle że polecenie zostanie uruchomione w katalogu w którym znajduje się wyszukany plik

- `du [opcje] ścieżka1 [ścieżka2 [...]]` wyświetlanie informacji o zajętej przestrzeni dyskowej przez wskazane pliki / katalogi, do ważniejszych opcji należy zaliczyć:
  `-s` podaje łączną ilość zajętego miejsca dla każdego argumentów (zamiast wypisywać rozmiar każdego pliku)
  `-c` podaje łączną ilość zajętego miejsca dla wszystkich argumentów
  `-h` stosuje jednostki typu k, M, G
  podawany rozmiar może się różnić (w obie strony) od wyniku ls: ls podaje rozmiar pliku (ile zawiera informacji lub ile zostało zadekarowane że może jej zawierać), a du to ile zajmuje na dysku
- `df [opcje]` wyświetlanie informacji o zajętości miejsca na poszczególnych systemach plików

W ścieżkach i nazwach plików można korzystać z znaków uogólniających powłoki takich jak:

- `?` oznaczający dowolny znak
- `*` oznaczający dowolny (także pusty) ciąg znaków
- `[a-z AD]` oznaczający dowolny znak z wymienionych w zbiorze ujętym w nawiasach kwadratowych, zbiór może być definiowany z użyciem zakresów, np. a-z AD oznacza dowolną małą literę od a do z włącznie, spację, dużą literą A lub D
- `[!a-z]` oznaczający dowolny znak z wyjątkiem znaków wymienionych w podanym zbiorze, zbiór może być definiowany z użyciem zakresów, np. a-z oznacza dowolną małą literę od a do z włącznie

Ścieżki mogą być podawane także jako napisy ujęte w cudzysłowie pojedynczym (`'`, np. `'aaa bbb'`) lub podwójnym (`"`, np. `"aaa bbb"`) celem np. ochrony spacji w nich występujących. Oba typy cudzysłowów zabezpieczają przed rozwijaniem znaków uogólniających (zastępowaniem napisu ze znakami listą pasujących nazw / ścieżek). Cudzysłów pojedynczy (w odróżnieniu od podwójnego) zabezpiecza także przed interpretacją umieszczonych wewnątrz innych znaków specjalnych takich jak odwołania do zmiennych.

Warto zauważyć, iż w przypadku komendy `ls` znaki uogólniające muszą być rozwinięte przez powłokę. Natomiast w przypadku komendy `find` na ogół chcemy aby nie były rozwijane przez powłokę, ale interpretowane przez samą komendę `find`. W tym celu powinniśmy je zabezpieczyć przed rozwinięciem przy pomocy cudzysłowów.

Zauważ również, że jeżeli komanda `ls` w wyniku rozwinięcia znaków uogólniających dostanie jako argument ścieżkę do katalogu to wylistuje jego zawartość (zachowanie to zmienia opcja `-d`).

3.1.3 kopiowanie, przenoszenie, usuwanie, ...

- `cp [opcje] źródło1 [źródło2 [...]] cel` 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
  `-a` podobnie jak `-r`, dodatkowo zachowując atrybuty plików
  `-l` zamiast kopiować tworzy twarde dowiązania (hard links)
  `-s` zamiast kopiować tworzy linki symboliczne do plików
  `-f` nadpisywanie bez pytania
  `-i` zawsze pytaj przed nadpisaniem
- `ln źródło1 [źródło2 [...]] cel` tworzy link twardy do wskazanego pliku (lub plików) w wskazanej lokalizacji, w przypadku wskazania wielu plików źródłowych cel powinien być katalogiem, do ważniejszych opcji należy zaliczyć:

`-s` tworzy dowiązania symboliczne zamiast twardych
`-r` używa ścieżki względnej zamiast bezwzględnej przy tworzeniu dowiązań symbolicznych

- `mv [opcje] źródło1 [źródło2 [...]] cel` przenosi wskazane pliki / katalogi do wskazanej lokalizacji, w przypadku przenoszenia wielu plików cel powinien być katalogiem, do ważniejszych opcji należy zaliczyć:
  `-f` nadpisywanie bez pytania
  `-i` zawsze pytaj przed nadpisaniem
- `rm [opcje] ścieżka1 [ścieżka2 [...]]` usuwa wskazane pliki, do ważniejszych opcji należy zaliczyć:
  `-r` pozwala na na (rekursywne) kasowanie katalogów wraz z zawartością
  `-f` usuwanie bez pytania
  `-i` zawsze pytaj przed usunięciem
- `mkdir [opcje] ścieżka1 [ścieżka2 [...]]` tworzy wskazane katalogi, do ważniejszych opcji należy zaliczyć:
  `-p` pozwala na tworzenie całej ścieżki a nie tylko ostatniego elementu, nie zgłasza błędu gdy wskazany katalog istnieje

4 Zadania

Zadanie 4.0.1

Wyświetlić nazwy (mogą być wraz z pełną ścieżką) wszystkich plików i katalogów znajdujących się bezpośrednio w `/etc/` których druga litera to `a` natomiast trzecia to `p` lub `s`.
Wskazówka: to zadanie nie wymaga stosowania find.

Zadanie 4.0.2

Wyszukaj (rekurencyjnie) wszystkie pliki w katalogu `/etc/` zmodyfikowane w przeciągu ostatnich 48 godzin.

Zadanie 4.0.3

Utworzyć w katalogu `/tmp` linki symboliczne `ll1` i `ll2` wskazujące na `/etc/passwd` odpowiednio poprzez ścieżkę bezwzględną i względną.

Zadanie 4.0.4

Zmodyfikuj rozwiązanie zadania 4.0.2 tak aby wyświetlać szczegóły (w tym datę modyfikacji) dla wyszukanych plików.

5 Praca domowa

5.1 Linux w domu

Część prac domowych można rozwiązać „na sucho”, bądź korzystając z interpretatorów on-line. Jeżeli masz już na swoim komputerze jakiś system „unixowy”, czyli np. jakąś dystrybucję Linuxa, ewentualnie system z rodziny BSD, czy nawet[3] *macOS* to super – możesz korzystać z niego do odrabiania prac domowych i samodzielnego zgłębiania omawianych zagadnień. Jeżeli nie posiadasz takiego systemu to gorąco zachęcamy do jego instalacji.

Polecaną (ale najtrudniejszą) metodą jest standardowe zainstalowanie dystrybucji Linuxa na wydzielonej partycji i utworzenie tzw. „dual boot”. Możesz także korzystać z systemu w postaci *liveUSB*, czyli uruchamianego z zewnętrznego nośnika USB (do jego utworzenia można skorzystać np. z https://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/) lub w postaci obrazu jakieś dystrybucji uruchamianego w ramach maszyny wirtualnej (np. przy pomocy *virtualbox* – https://www.virtualbox.org/ i https://www.osboxes.org/virtualbox-images/). Jeżeli nie wiesz na jaki system i jaką dystrybucję się zdecydować autor tego skryptu poleca zainstalować *Debian GNU/Linux* (ale to tylko jego subiektywna opinia ☺).

5.2 Instrukcja wysyłania rozwiązań

Rozwiązania zadań domowych należy przesłać na adres licealisci.pracownia@icm.edu.pl wpisując jako temat wiadomości `g2.x PD1`, gdzie `x` to numer grupy, np. `g2.1 PD1` 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 2020-03-03 godz. 16^{00}.

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.

3. Należy mieć na uwadze że *macOS* jest trochę odmienny od innych systemów „unixowych”, w szczególności od systemów z rodziny Linux na których oparty jest ten kurs.

5.3 Zadania domowe

Zadanie 5.3.1 — 1 pkt

Napisz polecenie, które utworzy katalog `/tmp/a/b` (katalog `b` znajdujący się wewnątrz katalogu `a` wewnątrz `/tmp`)

Zadanie 5.3.2 — 2 pkt

Napisz polecenie, które skopiuje pliki i katalogi (wraz z zawartością), znajdujące się w katalogu `/etc/`, których nazwa rozpoczyna się na literą `b` do katalogu `/tmp/`.

Zadanie 5.3.3 — 3 pkt

Napisz polecenie, które wyświetli nazwy plików znajdujących się bezpośrednio w katalogu `/etc/` (nie w jego podkatalogach), których rozmiar jest większy niż 300 bajtów.
Wskazówka: W skrypcie podane są tylko wybrane opcje omawianych poleceń. Rozwiązanie tego zadania może ułatwić zapoznanie się z pełną dokumentacją poleceń takich jak ***`ls`****, czy* ***`find`****.*