Podstawowy warsztat informatyka — lista 3

Zadanie 1. (0 punktów) **Rozwiązanie tego zadania jest niezbędne**, aby rozwiązać zadania z kolejnych list. Termin na wykonanie jest w skosie. **Aktualizacja 16-10-2024**: To zadanie zostało przeniesione na koniec listy, bo konfliktowało nieco z zadaniem 4.

Zadanie 2. (1 punkt) W nowym katalogu wykonaj polecenie

```
seq -w 0 10 1000 | sed 's/^/plik/' | xargs touch
```

Na drugiej stronie tej listy znajdziesz wyjaśnienie chata GPT odnośnie tego, co to polecenie robi.

Następnie poleceniem `rm` usuń pliki, których przedostatni znak nazwy to `3`.

Zadanie 3. (1 punkt) Zobacz, co jest w pliku `/etc/passwd`. Następnie zobacz, jacy użytkownicy są obecnie zalogowani do systemu, a jacy byli ostatnio. Wyszukaj, kiedy komputer był restartowany.

Zadanie 4. (2 punkty) *Uwaga: to zadanie należy wykonywać na pracowni – w domu prawdopodonie nie macie serwerów ssh.*

Połącz się przez `ssh` z serwerem `localhost`. Zauważ, że wymagało to wpisania hasła.

Wygeneruj poleceniem `ssh-keygen -t ecdsa` parę kluczy: prywatny oraz publiczny. Przy generowaniu, wybierz domyślną lokalizację i nie wpisuj hasła (czyli na wszystkie pytania odpowiadaj wciskając enter).

Wyświetl zawartość katalogu `.ssh` w Twoim katalogu domowym.

Wykonaj polecenie `ssh-copy-id localhost`. Następnie połącz się z serwerem `localhost` - czy wymagało to podania hasła? Spróbuj się zalogować jakiś inny włączony komputer w pracowni (np. `ssh lab110-20` — trzeba wybrać włączony komputer) – czy to wymagało hasła? Czemu?

Wyświetl zawartość katalogu `.ssh` w Twoim katalogu domowym. Pojawił się nowy plik. Obejrzyj pliki `~/.ssh/known_hosts` i `~/.ssh/authorized_keys`. Porównaj zawartość pliku `~/.ssh/authorized_keys` i `~/.ssh/id_ecdsa.pub`.

Zadanie 5. (1 punkt) Wywołaj polecenie `apt-get` z parametrem `moo` tak, aby jego wynik trafił do pliku `moo.txt`. Zmień nazwę tego pliku na `oo`. Stwórz polecenie (być może trzeba będzie użyć dwóch komend połączonych `|`), które wypisze tylko pierwsze dwie linie tego pliku (`head`...). Następnie wykonaj polecenie, które będzie nieustannie wypisywać drugą linię tego pliku.
Wskazówka: `yes ‘pwd’`, nieustannie wypisuje bieżący katalog. Uwaga: ‘‘ to są te znaki, które są tam, gdzie tylda, nie mylić z pojedynczymi apostrofami. Żeby przerwać takie polecenie, trzeba wcisnąć ctrl+c.

Zadanie 6. (1 punkt) Napisz polecenie wypisujące wszystkie liczby między 900 a 2023, które są podzielne przez 17. *Wskazówka: Warto uważnie przeczytać odpowiedź czata na kolejnej stronie.*

Zadanie 7. (0 punktów) Na każdym komputerze[1], na którym będziesz rozwiązywać zadania z PWI, poleceniem `ssh-keygen -t ecdsa` wygeneruj parę kluczy: prywatny oraz publiczny. Tym razem klucz prywatny zabezpiecz hasłem. Wszystkie klucze **publiczne** (`.pub`) dodaj do jednego pliku o nazwie `authorized_keys`. Kluczy prywatnych nikomu nie pokazuj. Przetestuj, że ten plik działa (np. umieszczając go w `~/.ssh` i wykorzystując `ssh localhost`). Wyślij ten plik (bez żadnej kompresji, katalogów itd.) przez odpowiednie zadanie w skosie.

[1] przy czym wszystkie komputery na pracowniach traktujemy tu jako jeden komputer, bo mają wspólny dysk sieciowy

MI

Omów krok po kroku co robi polecenie seq -w 0 10 1000 | sed 's/^/plik/' | xargs touch

Polecenie `seq -w 0 10 1000 | sed 's/^/plik/' | xargs touch` jest stosunkowo krótkim skryptem powłoki (bash), który generuje pliki o nazwach "plikXXXX" (gdzie "XXXX" to liczby w zakresie od 0000 do 1000) i następnie tworzy te pliki za pomocą polecenia `touch`. Oto krok po kroku, co się dzieje w tym poleceniu:

1. `seq -w 0 10 1000`: Polecenie `seq` służy do generowania sekwencji liczb. Tutaj generuje ona sekwencję liczb od 0 do 1000, z krokiem 10. Opcja `-w` oznacza, że liczby będą wypełniane zerami na początku, aby uzyskać stałą szerokość, na przykład 0000, 0010, 0020 itd.
2. `|`: Jest to operator potoku, który przekazuje wynik generowania sekwencji do kolejnego polecenia.
3. `sed 's/^/plik/'`: Polecenie `sed` służy do przetwarzania tekstu. Tutaj używa się go do dodania prefiksu "plik" do każdej linii wygenerowanej przez polecenie `seq`. Wyrażenie `'s/^/plik/'` oznacza zastąpienie początku każdej linii tekstu (symbol `^` oznacza początek linii) ciągiem "plik".
4. `|`: Kolejny operator potoku, który przekazuje przetworzony tekst do kolejnego polecenia.
5. `xargs touch`: Polecenie `xargs` służy do przekazania danych z potoku jako argumentów do innego polecenia, w tym przypadku `touch`. Polecenie `touch` służy do tworzenia pustych plików. Dzięki `xargs`, każda linia przetworzonego tekstu (czyli nazwa pliku w formie "plikXXXX") jest przekazywana jako argument do polecenia `touch`, co powoduje utworzenie tych plików.

Podsumowując, to polecenie generuje pliki o nazwach "plikXXXX" (gdzie "XXXX" to liczby od 0000 do 1000) i tworzy je w bieżącym katalogu za pomocą polecenia `touch`.