L1: Środowiska Wirtualne

Wymagane flagi kompilacji: `-Wall -fsanitize=address,undefined`

Napisz program symulujący tworzenie środowiska wirtualnego oraz zarządzania pakietami w środowisku wirtualnym. Po poprawnym wykonianiu program zwróci status `EXIT_SUCCESS`.

Tworzenie środowiska będzie polegało na:

- Utworzeniu folderu w bieżącym katalogu o nazwie takiej jak nazwa środowiska.
- Dodaniu do utworzonego katalogu pliku `requirements` (początkowo pustego).

Tworzenie wirtualnego środowiska realizowane jest za pomocą opcji `-c`. Opcja `-v` oznacza środowisko, na którym przeprowadzimy operację (w tym wypadku je tworzymy). Przykładowe wywołanie programu: `./prog1 -c -v <NAZWA_ŚRODOWISKA>`.

Zarządzanie środowiskami będzie polegało na instalowaniu i usuwaniu pakietów ze środowiska. Instalacja reprezentowana jest przez:

- Dopisanie linii z nazwą pakietu oraz po spacji jego wersji w pliku `requirements`. Pakiet może mieć dowolną nazwę i na ten moment musi mieć podaną dowolną wersję.
- Utworzenie pliku z nazwą pakietu, wypełnionego losową zawartością i uprawnieniami tylko do odczytu dla wszystkich klas użytkowników.

Deinstalacja odbywa się przez usunięcie linii z pakietem i pliku z nazwą pakietu. Składnia poleceń:

- Instalacja: `./prog1 -v <NAZWA_ŚRODOWISKA> -i <NAZWA_PAKIETU>==<WERSJA>`
- Deinstalacja: `./prog1 -v <NAZWA_ŚRODOWISKA> -r <NAZWA_PAKIETU>`

Program powinien znajdować środowiska w bieżącym katalogu. Opcja `-v` może być podawana wielokrotnie (z wyjątkiem przypadku tworzenia środowiska), co oznacza jednakową operację na różnych środowiskach (instalacja lub deinstalacja). Błąd na jednym środowisku nie powinien zatrzymać działania programu. Opcje mogą być użyte w różnej kolejności!

Dodatkowo program powinien poprawnie obsługiwać błędy (wypisywać komunikat i zwracać odpowiedni status) związane ze składnią poleceń, operacjami na plikach, dodawaniem, usuwaniem pakietów czy działaniem na nieistniejących środowiskach. Nie można dodać dwóch takich samych pakietów, nawet z inną wersją, oraz usuwać niedodanych.

Etapy:

1. 4 p. Implementacja tworzenia środowisk
2. 4 p. Implementacja instalacji pakietów i obsługa opisanych błędów
3. 4 p. Możliwość wielokrotnego używania opcji `-v`
4. 4 p. Implementacja usuwania pakietów

Etap	1	2	3	4	Suma
Punkty	4	4	4	4	16
Wynik					

Przykłady użycia programu:

```
$ ../prog1 -c nowe_srodowisko
$ ls ./
nowe_srodowisko
$ ls nowe_srodowisko/
requirements
```

Powyższe polecenie utworzy katalog `nowe_srodowisko`, a w nim plik `requirements`.

```
$ ../prog1 -v nowe_srodowisko -i numpy==1.0.0
$ export pandas=1.2.0
$ ../prog1 -v nowe_srodowisko -i pandas
$ cat nowe_srodowisko/requirements
numpy 1.0.0
pandas 1.2.0
$ ls -l nowe_srodowisko/
-r--r--r--  1 user  user   20 Oct 13 19:51 numpy
-r--r--r--  1 user  user   30 Oct 13 19:51 pandas
-rw-r--r--  1 user  user  17 Oct 13 19:50 requirements
$ ../prog1 -v nowe_srodowisko -r numpy
$ cat nowe_srodowisko/requirements
pandas 1.2.0
$ ls -l nowe_srodowisko/
-r--r--r--  1 user  user   30 Oct 13 19:51 pandas
-rw-r--r--  1 user  user  17 Oct 13 19:50 requirements
```

Powyższe polecenie zainstaluje dwa pakiety w `nowe_srodowisko`, w tym jeden z wcześniej zdefiniowaną wersją, a następnie usunie jeden z nich.

```
$ ../prog1 -v nieistniejace_srodowisko -i pandas
prog1: the environment does not exist
    usage: ...
```

Powyższe polecenie na nieistniejącym środowisku spowoduję błąd i wypisanie tak zwanego `usage`.