

Ćwiczenie 3

Skrypty powłoki

3.1. Tworzenie skryptów



1. Ikona  uruchomić program *Visual Studio Code*. Utworzyć nowy katalog o nazwie `cw3` w katalogu domowym, a nim zapisać ten tworzony plik nadając mu nazwę `pkt1`.
2. Zapisać pierwszą linię w pliku jako `#!/bin/bash`. Jest to instrukcja dla systemu uniksowego, aby uruchomić interpreter którego ścieżka i nazwa podana jest po znaku `!`.
3. Dodać drugą linię skryptu dopisując: `echo "Test polecenia skryptu"` w celu sprawdzenia czy utworzony skrypt zostanie uruchomiony. Zapisać plik (używając menu `File` `Save` albo skrótu klawiszowego `⌘ + S`).
4. Uruchomić terminal, poleceniem `cd` przejść do katalogu `cw3`. Wpisać polecenie `pkt1`, aby uruchomić zapisany skrypt. Dlaczego zamiast wyniku skryptu uzyskano odpowiedź: `bash: pkt1: command not found`?
5. Wpisać polecenie `./pkt1` aby uruchomić skrypt w bieżącym katalogu. Dlaczego zamiast wyniku skryptu uzyskano odpowiedź: `bash: ./pkt1: Permission denied`?
6. Wydać polecenie do zmiany uprawnień skryptu: `chmod 744 pkt1`.
7. Uruchomić skrypt przez wydanie polecenia `./pkt1`. Jaki jest wynik jego działania?

3.2. Polecenia warunkowe

8. Utworzyć nowy skrypt zapisując pod nazwą `pkt2` w katalogu `cw3`. Wpisać mu pierwszą linię interpretera i zmienić uprawnienia podobnie jak wcześniej.
9. W skrypcie `pkt2` użyć polecenia `test` w celu sprawdzenia czy plik `pkt1` istnieje. Jeżeli tak, to wyświetlić komunikat `Plik pkt1 istnieje`. W konsoli uruchomić utworzony skrypt `pkt2`.
10. Do skryptu `pkt2` dopisać linię, która sprawdzi czy plik `dane1` istnieje, a jeżeli nie istnieje to utwórz plik o takiej nazwie z zawartością `Linia tekstu w danych1`

3.3. Zmienne

11. Utwórz skrypt `pkt3`, który wczyta dwie liczby wprowadzone z klawiatury, a następnie wyświetli informację, która z nich jest większa.
12. Rozbuduj skrypt z poprzedniego punktu, aby wypisywał poprawną informację, gdy wprowadzone wartości są równe.
13. Sprawdź jak zachowa się utworzony skrypt, gdy zostanie podana tylko jedna liczba oraz gdy nie podano żadnej liczby. Dodaj zabezpieczenie, które nie pozwoli kontynuować skryptu, gdy nie podano wszystkich wartości. **Wskazówka:** przerwanie kontynuacji skryptu można wymusić poleceniem `exit` z opcjonalnym parametrem w postaci kodu zakończenia.

3.4. Polecenia iteracyjne

14. Utworzyć skrypt `pkt4`. W którym należy wykorzystać instrukcję pętli `for`, której zdaniem będzie rozpoznanie czy nazwy w bieżącym katalogu są nazwami plików, urządzeń albo katalogów. **Wskazówka:** Urządzenia znajdują się w katalogu `/dev`, stąd można przejść tam poleceniem `pushd /dev` i stamtąd uruchomić skrypt poleceniem `~/cw3/pkt4`.
15. Utworzyć skrypt `pkt5` z pętlą `while`, który wczyta tekst do wyświetlenia oraz liczbę, a następnie wyświetli ten tekst zadaną liczbę razy).
16. Utwórz skrypt `pkt6` z instrukcją `case` i pętlą `while`, który w zależności od wprowadzonego przez użytkownika tekstu podejmie następującą czynność:
 1. Gdy wprowadzono tekst `a`, to utworzy pusty plik o nazwie `daty.log` poleceniem `cat /dev/null > daty.log`.
 2. Gdy wprowadzono tekst `b`, to bieżąca data i godzina z polecenia `date` jest dołączona do pliku `daty.log`.
 3. Gdy wprowadzono tekst `c`, to wypisuje na ekranie zawartość pliku `daty.log`
 4. Gdy wprowadzono tekst `x`, to zakończy działanie skryptu.
 5. W pozostałych przypadkach wartość traktowana jest jak liczba, która mówi ile razy wypisać na ekran bieżącą datę i godzinę.

Przykładowe użycie tego skryptu: `a`

`b`
`b`
`b`
`c`
`5`

17. **Dla zaawansowanych.** Utworzyć skrypt, który wyjście polecenia `date` prześle do polecenia `read`, aby wyświetlić datę w formacie `rok-miesiac-dzien_godzina:minuty:sekundy`. Nie można w tym celu użyć parametru `"%Y-%m-%d %H:%M:%S"` polecenia `date`.

3.5. Obsługa edytora VI / VIM

18. Przejść do katalogu `cw3`. Otworzyć plik `pkt1` w edytorze VIM poleceniem `vi pkt3`.

19. Przetestować poruszanie się po pliku w trybie poleceń klawiszami `j` linia w dół, `k` linia w górę, `l` znak w prawo, `h` znak w lewo.
20. Wrócić na początek pliku poleceniem `gg`.
21. Rozpocząć wyszukiwanie tekstu *echo* w pliku poleceniem `/echo` `↵`.
- Nacisnąć `n` do wyszukiwania kolejnego wystąpienia.
 - Kolejny raz nacisnąć `n`, aby przejść do następnego wystąpienia słowa *echo*.
 - Kolejne naciśnięcie `n` przeskoczy do początku pliku wyświetlając w pasku informację, że wyszukiwanie osiągnęło koniec pliku i rozpoczęło się od początku.
 - Nacisnąć `2n` aby przejść o dwa wystąpienia dalej.
 - Nacisnąć `N`, aby przejść do poprzedniego wyszukiwania.
22. Przejść do ostatniej linii w pliku poleceniem `G`.
23. Usunąć bieżącą linię poleceniem `dd`.
24. Dodać nową linię poniżej stosując polecenie `o`.
25. Wprowadzić dowolny tekst komentarza np. `# to jest linia komentarza`, wyjść z trybu wprowadzania klawiszem *Escape* (`⌫`).
26. Wstawić wyciętą linię poleceniem `p`.
27. Zapisać plik poleceniem `w` `↵`.
28. Przejść do linii nr 3 poleceniem `3G`.
29. Używając polecenia `i` wstawić przed `read` następujący tekst `IFS=,␣`.
30. Opuścić tryb wstawiania klawiszem *Escape* i zapisać plik poleceniem `w`.
31. Uruchomić skrypt poleceniem `!./pkt3` `↵`. Jako wartości należy podać liczby oddzielone przecinkiem, np. `2,3`.
32. Zakończyć pracę edytora VIM przez `q` `↵`.

3.6. Zadania dla ambitnych

33. Napisz skrypt w Bash, który monitoruje wykorzystanie procesora (CPU) oraz ilość pamięci zajmowanej przez pliki w pamięci podręcznej na systemie macOS, zapisując wyniki wraz ze znacznikiem czasu co sekundę. Skrypt powinien działać w nieskończonej pętli, aż do ręcznego zakończenia, i wyświetlać ilość pamięci podręcznej w gigabajtach (GB) z dokładnością do dwóch miejsc po przecinku. Dodatkowo, skrypt powinien obsługiwać przerwanie działania (`Ctrl+C`) i wyświetlać odpowiedni komunikat o zakończeniu monitorowania.

- Użyj polecenia `top`, aby uzyskać aktualne obciążenie procesora w procentach. Przetwórz wynik tak, aby wyodrębnić wartość procentową wykorzystania CPU.
- Użyj polecenia `vm_stat`, aby uzyskać liczbę stron pamięci zajmowanych przez pliki w pamięci podręcznej (`File-backed pages`) oraz liczbę stron pamięci, które mogą być oczyszczone (`Pages purgeable`).

- Oblicz sumę wartości `File-backed pages` i `Pages purgeable`, a następnie przelicz ją na gigabajty (GB). Przyjmij, że jedna strona pamięci ma rozmiar 4096 bajtów (4 KB). Obliczenie pamięci w gigabajtach wykonaj według wzoru:

$$\text{cached_memory_in_gb} = \frac{(\text{File-backed pages} + \text{Pages purgeable}) \times 4096}{1024^3}$$

- Wyświetl informacje o aktualnym czasie, obciążeniu CPU oraz ilości pamięci zajmowanej przez pliki w pamięci podręcznej, wyrażonej w GB.
- Skrypt powinien obsługiwać sygnał przerwania (Ctrl+C) i wyświetlać komunikat o zakończeniu monitorowania wraz z datą i czasem zakończenia.

34. Napisz skrypt `Bash`, który symuluje zapełnianie pamięci RAM poprzez tworzenie tymczasowych plików o zadanej wielkości. Skrypt powinien działać według następujących założeń:

- Skrypt powinien domyślnie tworzyć bloki o rozmiarze 100 MB, przechowywane w katalogu tymczasowym.
- Skrypt przyjmuje jeden argument, który określa łączną ilość pamięci (w MB), którą należy zapełnić.
- Skrypt powinien informować użytkownika o każdej kolejnej zaalokowanej części pamięci, podając bieżącą wielkość przydzielonej pamięci.
- Skrypt powinien posiadać funkcje do usuwania plików i zwalniania pamięci.
- Po wykryciu sygnału `SIGINT` (Ctrl+C), skrypt powinien usunąć wszystkie utworzone pliki i zwolnić zajętą pamięć, wyświetlając odpowiedni komunikat.
- Po zapełnieniu określonej ilości pamięci, skrypt powinien przechowywać dane w pamięci aż do momentu, kiedy użytkownik przerwie działanie skryptu za pomocą `Ctrl+C`.

3.7. Zakończenie ćwiczenia

35. Usunąć utworzone pliki i katalogi poleceniem: `rm -rf ~/cw3`.