[SOP] Komendy w Bash'u:

`#!/bin/bash` ← rozpoczyna nowy skrypt

`touch file.txt` ←tworzy nowy plik

`ls` ← wyświetlić zawartość katalogu

`ls -l` ← wyświetlić zawartość katalogu w długim formacie

`cat file.txt` ← łączy i drukuje pliki

`cat -n file.txt` ← numeruje linie wyjściowe zaczynając od 1

`wc file.txt` ← liczy liczbę słów, wierszy, znaków i bajtów

`wc -c file.txt` ← liczy liczbę bitów

`wc -l file.txt` ← liczy liczbę linii

`wc -m file.txt` ← liczy liczbę znaków

`wc -w file.txt` ← liczy liczbę słów

`head file.txt` ←wyświetla pierwsze linie pliku

`head -n 5 file.txt` ←wyświetla 5 pierwszyc linii pliku

`tail file.txt` ←wyświetla ostatnie linie pliku

`tail -n 5 file.txt` ←wyświetla ostatnie 5 linii pliku

`sort file.txt` ←sortuje lub łączy rekordy (wiersze) plików tekstowych

`sort -R file.txt` ← sortuj według losowej kolejności

`sort -r file.txt` ← sortuje w odwrotnej kolejności

`uniq file.txt` ← odfiltrowuje powtarzające się wiersze w pliku

`uniq -c file.txt` ← każda linia wyjściowa poprzedzona jest liczbą wystąpień danej linii na wejściu, po której następuje pojedyncza spacja.

`uniq -d file.txt` ← wyświetla pojedynczą kopię każdej linii, która powtarza się w danych wejściowych

`uniq -i file.txt` ← porównywanie wierszy bez uwzględniania wielkości liter

`uniq -u file.txt` ← wyświetla tylko te linie wyjściowe, które nie powtarzają się na wejściu

`find` ← chodzić po hierarchii plików (`find .` , `find *`)

`find ./ -type f -name 't*.txt'` ← znajdź plik typu plik o nazwie 't*.txt'

`cut` ← wycina wybrane fragmenty każdego wiersza pliku

`cut -b list file.txt` ← lista określa pozycje bajtów

`cut -c list file.txt` ← lista określa pozycje znaków

`cut -f list file.txt` ← lista określa pola, oddzielone w danych wejściowych przez pole znak ograniczający

`tr string1 string2` ← przetłumaczy znaki

`tr -c string1 string2` ← uzupełnia zestaw wartości w string1

`tr -d string1 string2` ← usuwa znaki w string1

`nl file.txt` ← filtr numeracji linii

`grep pattern file.txt` ← wyszukiwarka wzorców plików

`egrep pattern file.txt` ← wyszukiwarka wzorców plików

`test, [ ]` ← narzędzie do oceny warunków

- `-f file` ← jeśli istnieje i jest zwykłym plikiem
- `-d file` ← jeśli istnieje i jest katalogiem
- `-e file` ← `plik` istnieje
- `-r file` ← `plik` istnieje i przyznano uprawnienia do odczytu
- `-w file` ← `plik` istnieje i przyznano uprawnienia do zapisu
- `-x file` ← `plik` istnieje i przyznano uprawnienia do wykonywania
- `file1 -ot file2` ← `plik1` jest starszy niż `plik2`
- `file1 -nt file2` ← `plik1` jest nowszy niż `plik2`
- `-z string` ← długość `string` wynosi zero
- `-n string` ← długość `string` jest niezerowa
- `wyrażenie1 -a wyrażenie2` ← zarówno `wyrażenie1` jak i `wyrażenie2` są prawdziwe
- `wyrażenie1 -o wyrażenie2` ← albo `wyrażenie1` albo `wyrażenie2` ma wartość true
- `string1 = string2` ← ciągi są równe
- `string1 != string2` ← ciągi nie są równe
- `int1 -eq int2` ← `int1` jest równy `int2`
- `int1 -ge int2` ← `int1` jest większy lub równy `int2`
- `int1 -gt int2` ← `int1` jest większy niż `int2`
- `int1 -le int2` ← `int1` jest mniejsza lub równa `int2`
- `int1 -lt int2` ← `int1` jest mniejszy niż `int2`
- `int1 -ne int2` ← `int1` nie jest równy `int2`

`cat fruit.txt | grep App.e` ← dopasowuje **dowolny znak** oprócz nowej linii

`cat fruit.txt | grep ^B` ← dopasowuje **początek łańcucha**

`cat fruit.txt | grep e$` ← dopasowuje **koniec łańcucha**

`cat fruit.txt | grep ap*le` ← dopasowuje zero lub więcej wystąpień, czyli **dowolną liczbę razy, znaku w łańcuchu**

`cat fruit.txt | grep "\ "` ← używane do **unikania następującego po nim znaku**

`cat fruit.txt | grep -E "(fruit)"` ← używane do **dopasowywania lub wyszukiwania zestawu wyrażeń regularnych**

`cat fruit.txt | grep -E Ch?` ← dopasowuje dokładnie **jeden znak w łańcuchu** lub strumieniu

`chmod`

`cd`

`nano`

`man`

`clear`

`file`

`more`

`less`

`let`

`read`

`read -r`

`mv`

`cp`

`rm`

`rm -f`

`rm -r`

`rmdir`

`pwd`

`rename`

`let x=5+4`

`` `expr x + 5` ``

`x=$[5 + 4]`

`$(())` ← operacje arytmetyczne w subshell'u

`$0` ← argument z nazwą skrytpu i scieżką

`$#` ← liczba występujacyhc w tablicy parametrów

`$@` ← zwraca całą zawartość aktualnej tablicy

`*` reprezentuje jeden lub wiele znaków

`?` reprezentuje dokładnie jeden znak

`[abcde]` reprezentuje dokładnie jeden znak z wymienionych

`[a-e]` reprezentuje dokładnie jeden znak z przedziału

`[!abcde]` reprezentuje dowolny niewymieniony znak

`[!a-e]` reprezentuje znak nienależący do przedziału

`{plik,file}` reprezentuje dowolny z wymienionych ciągów

`.bmp` reprezentuje wszystkie pliki kończące się na .bmp

`?[0-9][a-d]` reprezentuje plik o trzyznakowej nazwie, gdzie pierwszy znak jest dowolny, drugi jest cyfrą, zaś trzeci literą z przedziału a-d

`{nowy,new}*.txt` reprezentuje wszystkie pliki zaczynające się od nowy lub new będące plikami txt

---

1. Utwórz katalog o nazwie “SOP” w katalogu domowym.
2. Przejdź do katalogu “SOP” i utwórz katalog “Zajecia1”.
3. Przejdź do katalogu domowego i bezpośrednio, za pomocą jednej komendy przejdź do katalogu “Zajecia1”.
4. Utwórz dwa pliki, “Test1.txt” i “Test2.txt”.
5. Zmień uprawnienia pliku “Test2.txt” na czytanie i wykonanie. Sprawdź czy uprawnienia zostały zmienione.
6. Przejdź do katalogu domowego i wyświetl wszystkie katalogi wraz z ich zawartością.
7. Zapisz do pliku “Test1.txt” tekst “Hello Linux”, a następnie go wyświetl.
8. Dodaj do pliku “Test1.txt” co najmniej 15 linii tekstu.
9. Wyświetl posortowaną zawartość pliku “Test1.txt”.
10. Wyświetl pierwsze 4 linie pliku “Test1.txt”.
11. Wyświetl ostatnie 3 linie pliku “Test1.txt”.
12. Wyświetl liczbę linii pliku “Test1.txt”.
13. Wyświetl liczbę słów w pliku “Test1.txt”.
14. Utwórz katalog “Zajecia2”. Przygotuj plik, który zawiera co najmniej 30 linii, z czego niektóre linie powtarzają się, jedna po drugiej.
15. Utwórz plik “plik2.txt” i zapisz do niego pierwsze 10 linii z wcześniej przygotowanego pliku za pomocą strumienia.
16. Dopisz do końca pliku “plik2.txt” ostatnie 10 linii wcześniej przygotowanego pliku za pomocą strumieni.
17. Wyświetl zawartość pliku “plik2.txt” na konsoli za pomocą komendy echo i strumieni.
18. Utwórz plik “Bledy.txt”, następnie dopisz na koniec pliku 5wygenerowanych błędów.

19. Wyświetl pierwszy plik na konsoli, pomniejszony o powtarzającesię linie.(użycie potoków)
20. Posortuj linie z pliku “Bledy.txt” i wyświetl je na konsoli za pomocą potoków.
21. Utwórz i uruchom skrypt bash, który wyświetli na konsoli napis “Hello Linux”.
22. Utwórz i uruchom skrypt bash, który wyświetli na konsoli sumę dwóch podanych liczb, jako argumenty.
23. Utwórz i uruchom skrypt bash, który wyświetli wszystkie przekazane argumenty i ich ilość.
24. Utwórz skrypt, który przyjmuję nazwę pliku jako argument. Skrypt ma posortować linie, usunąć nieunikalne linie i zapisać zmiany do pliku “result1.txt”.
25. Utwórz skrypt, który wyszuka w katalogu domowym plik, którego nazwa została podana w argumencie. Następnie wypisz ścieżkę na konsoli.
26. Utwórz skrypt, który dopiszę wszystkie linie zaczynające się na literę “a”, z pliku podanego w pierwszym argumencie, do pliku, którego nazwa to drugi argument.
27. Napisz skrypt, który policzy ilość samochodów w salonie po określonej liczbie dni. Codziennie w salonie sprzedawana jest połowa wszystkich samochodów, które aktualnie są na stanie. Ilość dni i samochodów to argumenty wejściowe.
28. Napisz skrypt, który wyświetli linie, które zaczynają się na “a” i zawierają co najmniej 10 znaków ze wszystkich plików tekstowych w bieżącym katalogu.
29. Utwórz skrypt, który wybierze 3, 4 i 5 linie z każdego pliku tekstowego, posortuje, usunie duplikaty i zapisze wraz z numerem linii do pliku, którego nazwa została przekazana jako argument.
30. Napisz skrypt, który będzie przyjmować listę argumentów, gdzie każdy argument może oznaczać jakieś zwierzę. Skrypt ma za zadanie wypisać na konsoli nazwę zwierzęcia wraz z dźwiękiem jakie wydaje (np. Kot robi miau). Ilość rodzajów zwierząt rozpoznawanych przez skrypt nie powinna być mniejsza niż 3. W przypadku zwierzęcia nie obsługiwanego wyświetlamy komunikat jakie zwierzęta obsługujemy.
31. Napisz skrypt, który usunie wszystkie pliki tekstowe z katalogu bieżącego, które zawierają więcej niż 10 linii.
32. Napisz skrypt, który przeniesie wszystkie pliki tekstowe, które nazwy zaczynają się od “t”, z bieżącego katalogu i subkatalogów do katalogu podanego jako pierwszy argument. Wszelkie błędy przekieruj do pliku errors.txt.
33. Napisz skrypt, który znajdzie najdłuższą linię, zaczynającą się na literę “c”, w pliku podanym jako pierwszy argument. Następnie zapisz tą linie do pliku, którego nazwa to drugi argument.

34. Napisz skrypt, który zmodyfikuje wszystkie pliki z rozszerzeniem .log w katalogu domowym. Jeśli w pliku jest więcej niż 10 linii, zostawiamy ostatnie 10 i wypisujemy na konsoli nazwę pliku/ścieżkę wraz z komunikatem o redukcji. Jeśli jest mniej lub równo wypisujemy informację o zbyt małej ilości linii do zredukowania.
35. Napisz skrypt, który wyświetli nazwę z najnowszego pliku w katalogu bieżącym.
36. Napisz skrypt, który przejdzie po wszystkich plikach tekstowych w katalogu domowym. Sprawdź, czy plik ma uprawnienia do odczytu, jeśli nie to je nadaj. Następnie odczytaj linie z zakresu <5,8>, posortuj je, usuń powtarzające się linie i policz ilość znaków. Policzoną wartość dla każdej linii sumuj i wyświetl na konsoli.
37. Napisz skrypt, który jako pierwszy argument przyjmuję plik. Sprawdź, czy plik istnieje i jest regularny, a następnie przejdź po liniach w zakresie <2,4> i wyświetl każdy znak wraz z jego indeksem w linii.
38. Napisz skrypt, który przyjmuje nazwę pliku jako pierwszy argument. Jeśli plik istnieje i ma uprawnienia do odczytu, wyświetl ostatnie 7 osortowanych linii wraz z liczbą ich wystąpień w pliku. Jeśli plik nie istnieje lub nie ma uprawnień, wypisz informację na konsoli.
39. Napisz skrypt, który przyjmuję nazwę pliku jako pierwszy argument. Policz, ile plików jest nowszych, a ile starszych w bieżącym katalogu i wyświetl te informację na konsoli.
40. Napisz skrypt, który przyjmuję listę nazw plików jako argumenty. Jeśli plik istnieje, znajdź wszystkie linie zaczynające się od “error” oraz mają co najmniej 3 słowa. Następnie znajdź słowa w tych liniach, które odwrócone (początek z końcem) są pierwszym i ostatnim słowem w leksykograficznym porządku. Następnie zapisz te słowa do pliku “reversed.txt”.
41. Napisz skrypt, który wybierze pięć najmniejszych plików w bieżącym katalogu i wypisze informację, czy każdy z nich jest plikiem czy katalogiem.
42. Napisz skrypt, który przyjmie nazwę katalogu jako pierwszy argument. Jeśli katalog istnieje i ma uprawnienia do odczytu, przeszukaj ten katalogi i wszystkie podkatalogi w poszukiwaniu pliku, którego nazwa została podana jako drugi argument. Jeśli plik ma uprawnienia od edycji, usuń ten plik i wyświetl informację o jego usunięciu wraz z nazwą tego pliku.
43. Napisz skrypt, który przejdzie po wszystkich plikach tekstowych w bieżącym katalogu i wszystkich podkatalogach. Jeśli masz uprawnienia do edycji pliku, zapisz do niego linie, gdzie każda kolejna będzie dłuższą sekwencją małych liter, zaczynając od a, do z.
44. Wyświetl nazwę i zawartość roboczy katalogu użytkownika.

45. Przejdź do katalogu nadrzędnego swojego katalogu domowego i sprawdź odpowiednim poleceniem jaka jest ścieżka dostępu do tego katalogu i wyświetl zawartość. Co się stało? Czemu nie widzisz jego zawartości? Użyj polecenia ls i jego opcji l i d aby poznać powód. Odpowiedzią jest pierwsza kolumna, tylko trzeba umieć ją odpowiednio odczytać.
46. Wyświetl pomoc do polecenia ls i korzystając z niego wyświetl WSZYSTKIE pliki z katalogu /bin wraz z ich tzw. inodami. Co to jest inode (iwęzeł) ? Wykorzystaj opcje plecenia ls. Wykonaj to polecenie na dwa sposoby:
    a. pierwszy sposób: wyświetl zawartość katalogu /bin będąc w tym katalogu (używając polecenia cd przejdź do tego katalogu, a następnie wyświetl jego zawartość)
    b. drugi sposób: wyświetl zawartość katalogu /bin będąc we własnym katalogu domowym
47. Wszystkie pliki z katalogu /bin wyświetl w formacie długim, tzn. z datą, wielkością i powinny być:
    a. w wersji 1: posortowane względem wielkości od najmniejszych do największych.
    b. w wersji 2: posortowane względem dary modyfikacji od najnowszych do najstarszych
48. Przejdź do katalogu domowego i utwórz pusty katalog o nazwie SOP a w nim plik o nazwie ala_ola.txt. Do pliku wpisz dwa zdania (każde w osobnej linii): Ala ma 1 kota. Ola ma 2 psy. Wyświetl ten plik od początku do końca i od końca do początku.
49. W katalogu głównym / utwórz podkatalog Kula. Nie udało się :-) Ciekawe dlaczego? Kto ma prawo zapisu do tego katalogu, jak to odczytać? Podobne do zadania 4, ale trochę trudniejsze.
50. Wróć do katalogu SOP w swoim katalogu domowego i w nim utwórz dwa katalogi Kula i Kula1.
51. W katalogu Kula utwórz podkatalog Kula2. Do podkatalogu Kula2 skopiuj plik ala01.txt.
52. Cały katalog Kula z zawartością skopiuj do katalogu Kula1, a nazwę pliku ala01.txt zmień na ala02.txt. W katalogu SOP utwórz katalog Kuleczka i przenieś tam zarówno katalog Kula1 i Kula z ich zawartością.
53. W katalogu **/tmp** utwórz katalog o nazwie Twojego loginu. Skopiuj plik **/etc/passwd** do tego katalogu i nadaj mu nazwę OtoNaszeHasla.txt.
54. Do pliku OtoNaszeHasla.txt utwórz łącznik symboliczny (trochę odpowiednik skrótu z Windows), nazwij go PASS i umieść go w tym samym katalogu. Wykonaj polecenie "ls -il ~". Porównaj numery inode w oryginalnym pliku i w łączniku. Jak poznać, że

jakiś plik jest łącznikiem? Może użyj opcji -l polecenia ls i popatrz na pierwszy znak w linii odpowiadającej temu łącznikowi.

55. Wykorzystując polecenia more, less i cat wyświetl zawartość tego pliku, a w szczególności hasło użytkownika root. Zrób to wykorzystując łącznik PASS. Oczywiście hasła tam raczej nie znajdziemy.
56. Zmień nazwę pliku OtoNaszeHasla.txt na OtoNieNaszeHasla.txt i powtórz zadanie 14. Udało się? Dlaczego?
57. A teraz do pliku OtoNieNaszeHasla.txt utwórz łącze sztywne on nazwie szPASS. Zmień nazwę pliku OtoNieNaszeHasla.txt na OtoNaszeHasla.txt. Sprawdź czy po zmianie nazwy oryginalnego pliku możesz odwołać się do tego pliku poprzez łącznik szPASS. Czym jest w tym przypadku szPASS? Porównaj numery inode pliku szPASS i pliku OtoNaszeHasla.txt.
58. Znajdź w manualu, jak w less szuka się wzorca wstecz.
59. Wyczyść ekran.
60. Skasuj cały katalog Kuleczka z zawartością
61. Wyświetl wszystkie informacje o systemie, na którym pracujesz typ uname –a
62. Wyświetl „banner” ze swoim imieniem
63. Wyświetl pomoc do polecenia ls i korzystając z niego wyświetl WSZYSTKIE pliki z katalogu /bin wraz z ich tzw. i-nodami. Co to jest i-node?
64. W katalogu /tmp utwórz katalog o nazwie Twojego loginu. Skopiuj plik /etc/passwd do tego katalogu i nadaj mu nazwę OtoNaszeHasla.txt
65. Do pliku OtoNaszeHasla.txt utwórz łącznik symboliczny (odpowiednik skrótu z Windows), nazwij go PASS i umieść go w tym samym katalogu. Wykonaj polecenie "ls -il". Porównaj numery i-node w oryginalnym pliku i w łączniku. Jak poznać, że jakiś plik jest łącznikiem?
66. Wykorzystując polecenia more, less i cat wyświetl zawartość tego pliku, a w szczególności hasło użytkownika root. Zrób to wykorzystując łącznik PASS.
67. Zmień nazwę pliku OtoNaszeHasla.txt na OtoNieNaszeHasla.txt i powtórz zadanie 3. Udało się? Dlaczego?
68. A teraz do pliku OtoNieNaszeHasla.txt utw?z łącze sztywne on nazwie szPASS. Zmień nazwę pliku OtoNieNaszeHasla.txt na OtoNaszeHasla.txt. Sprawdź czy po zmianie nazwy oryginalnego pliku możesz odwołać się do tego pliku poprzez łącznik szPASS. Czym jest w tym przypadku szPASS? Porównaj numery i-node pliku szPASS i pliku OtoNaszeHasla.txt.
69. Zdefiniuj trzy zmienne IMIE i NAZWISKO i WIEK. Powinny one zawierać odpowiednie dane, czyli twoje imię i nazwisko i twój wiek.

70. Zapisz do pliku ABC.TXT całą zawartość katalogu /etc wraz z podkatalogami. Policz ile tam jest znaków i linii i taką informację zapisz do pliku o nazwie RESULT.TXT, który powinien zostać utworzony w katalogu sop, który będzie w twoim katalogu domowym.
71. Korzystając z potoków zamień wszystkie spacje na kropki w pliku ABC.TXT z poprzedniego zadania i wynik zapisz do tego samego pliku. Jak ktoś nie potrafi zapisać do tego samego pliku to niech zapisze do innego :-).
72. Na koniec pliku ABC.TXT dopisz plik RESULT.TXT. Przydatne wyrażenia i polecenia: cat i >>. Plik ABC.TXT skasuj
73. Dlaczego prawie na końcu ostatniego przykładu jest błąd expr: syntax error?
74. W zmiennej SUMA powinna się pojawić liczba wszystkich plików i katalogów jakie są w katalogach /bin, /usr/bin i ~. oraz liczba linii we wszystkich plikach java z Twojego katalogu domowego.
75. Napisz prosty skrypt, który wyświetli "Hello world".
76. Napisz skrypt który wyznaczy pierwiastki równania kwadratowego. Dla uproszczenia pierwiastek kwadratowy z liczby zapisanej w zmiennej LICZBA wyznaczamy w sposób następujący:
77. Napisz polecenie, który sprawdzi, czy dany ciąg znaków jest:
    a. liczbą wyłącznie dodatnią
    b. liczbą dodatnią lub ujemną
78. Napisz prosty skrypt, który:
    a. wyświetli zawartość katalogu /bin;
    b. policzy ile tam jest plików i katalogów i zapisze tę informację do zmiennej LICZBA_WSZYSTKICH;
    c. policzy ile tam jest plików i zapisze tę informację do zmiennej LICZBA_PLIKOW;
    d. wyświetli ilość katalogów;
79. Napisz skrypt, który wyświetli wszystkie pliki z katalogu /bin, które:
    a. zawierają w nazwie literę a.
    b. są do odczytu dla wszystkich.
    c. na końcu nazwy mają małą literkę s lub d.
    d. na końcu nazwy mają małą literkę.
80. Napisz skrypt, który wyświetli wszystkie pliki z katalogu /dev, które:
    a. są urządzeniami blokowymi.
    b. na końcu nazwy mają cyfrę lub dwie cyfry.

c. są urządzeniami blokowymi i na końcu nazwy mają cyfrę lub dwie cyfry.

d. w nazwie mają wyłącznie i dokładnie 3 dowolne znaki.

e. w nazwie mają nie więcej niż 3 dowolne znaki.

71. Napisz skrypt, który pobierze jako argument nazwę katalogu i:

    a. wyświetli nazwy wszystkie plików z tego katalogu (wyłącznie nazwy)

    b. policzy i wyświetli informacje ile tych plików tam jest

    c. policzy i wyświetli informacje ile tam jest katalogów

    d. policzy i wyświetli informacje ile tam jest w sumie katalogów i plików

72. Napisz skrypt0, który sam sobie nada uprawnienia do zapisu (o ile tych uprawnień nie miał) i zabierze uprawnienia do zapisu (jak tych uprawnienia miał).

73. Napisz skrypt1, które wypisze nazwy plików z całego katalogu, które zaczynają się od kropki, w nazwie posiadają literkę b. Nazwa katalogu zadana jest jako argument skryptu. Wyniki mają być zapisane do pliku o nazwie PLIK.

74. Napisz skrypt2, który wywoła skrypt1, jeżeli skrypt1 zakończy się sukcesem to skrypt 2 policzy ile jest plików i katalogów wygenerowanych przez skrypt 1.

75. Policz ile jest katalogów i podkatalogów w katalogu który jest pierwszym argumentem skryptu.

76. Napisz skrypt, który wypisze nazwy katalogów z katalogu głównego (/), które zaczynają się od liter b lub e lub c.

77. Napisz skrypt, który policzy ile plików z zadanymi rozszerzeniami jest na wybranym katalogu. Nazwa katalogu powinna być podana jako pierwszy argument, a rozszerzenia jako pozostałe argumenty.

78. Napisz skrypt, który z pliku /tmp/ekg2-current.tar.gz usunie wszystkie pliki tekstowe - z rozszerzeniem txt - i napisze ile tych plików tam było, oraz jaką sumaryczną liczbę linii one miały. Plik dostępny jest na maszynie msh. Oczywiście przed wykonaniem zadania skopiuj powyższy plik do katalogu domowego.

79. Napisz skrypt, w który wyświetla wszystkie swoje argumenty w odwrotnej kolejności

80. Napisz skrypt, w którym użytkownik podaje numer argumentu, który ma być wyświetlony na ekranie, a następnie wyświetlamy ten argument.

81. Napisz skrypt, który wyświetli liczbę linii w plikach z rozszerzeniem java w wybranym katalogu (argument skryptu). Dla każdego pliku wyznaczamy liczbę linii niezależnie.

82. Napisz skrypt, który zapisze do pliku środkowe linie ze wszystkich skryptów ze wskazanego katalogu (). Nazwa pliku podana jest jako argument. Linia środkowa to linia

33. Napisz skrypt, który wyświetli silnie z liczby a. , która jest wprowadzona z klawiatury. Liczba a musi być dodatnia
34. Napisz skrypt, który wyświetli wszystkie silnie od liczby a do liczby b. Liczby a, b muszą być dodatnie.
35. Napisz skrypt, który co kilka sekund sprawdza czy użytkownik o jakimś loginie zalogował się w systemie 2-krotnie. Jeżeli nie to skrypt zasypia na kilka sekund i sprawdza później. Jak użytkownik się zalogował 2-krotnie to, skrypt się kończy.
36. Napisz skrypt, który policzy ile masz plików z rozszerzeniem java na swoim dysku Z, a ile jest plików z rozszerzeniem class. Wyznacz ich różnicę.
37. Napisz skrypt, który sprawdzi, czy podany argument jest nazwą katalogu i jak tak to go wyświetli wraz ze wszystkimi podkatalogami i policzy rozmiar jego rozmiar w bajtach. Dodatkowo, nazwy wszystkich plików lub katalogów z ustawionym atrybutem wykonywania dla właściciela powinny zostać zapisane do pliku o nazwie, która zostanie podana jako drugi argument. Jeżeli drugiego argumentu nie ma to skrypt ma nic nie robić.
38. Mamy plik jak poniżej:

    1 2 3 4 5

    4 3 2 3 4 5 4 3

    2 2 2 2 22 3 34 4 4 5 5 5

    d 3 43 54 5 3

    Napisz skrypt, który pobierze nazwę takiego pliku jako pierwszy argument i wyznaczy sumę (opcja -s) tych liczb i/lub iloczyn (opcja -i) tych liczb.
39. Napisz skrypt, który wyświetli tabelkę jak mnożenia
90. Napisz skrypt (lub polecenie), który wypisze nazwy tych katalogów z katalogu bieżącego, w których nazwie znajduje się literka C.
91. Napisz skrypt (lub polecenie), który wypisze nazwy tych katalogów z katalogu bieżącego, którego nazwa składa się z przynajmniej 4 znaków.
92. Napisz skrypt (lub polecenie), który wypisze nazwy tych katalogów z katalogu bieżącego, które zwierają co najmniej jeden plik do odczytu.
93. Napisz skrypt (lub polecenie), który wypisze te spośród swoich argumentów, które są plikami zawierającymi w środku co słowo CPU. Z przodu słowa musi być spacja
94. Mamy plik o zawartości jak poniżej:

    1 2 3 4 5

    9 2 3 1 2 a l a i o l a

    9 2 ala ma kota 102

Napisz skrypt, który pobierze nazwę takiego pliku jako pierwszy argument i wyznaczy ilość wszystkich cyfr i liter w takim pliku. Plik oczywiście może mieć zmienną ilość linii i elementów w linii.

95. Utwórz i wyświetl, strukturę katalogow zgodną ciągiem liczb Fibonacciego, czyli w bieżącym katalogu tworzymy jeden katalog, w tym katalogu tworzymy też jeden katalog, w tym katalogu, tworzymy dwa katalogi, w tych dwóch tworzymy trzy, a w każdym z tych sześciu utworzonych tworzymy pięć katalogów. Utwórz to w możliwie najkrótszy sposób
96. Do powyższego rozwiązania dodaj jeszcze jeden poziom katalogów. Czy jesteś w stanie policzyć ile katalogów zostało utworzonych?
97. Najszybszą znaną sobie metodą utwórz 40 dowolnych katalogów. Skasuj potem te katalogi
98. W każdym z tych 40 katalogów, utwórz po 10 plików? Oczywiście zweryfikuj czy się udało to zrobić. Najlepiej aby te pliki zawierały jakąś treść, ale jeżeli nie potrafisz tego zrobić, to mogą być i puste, Po ćwiczeniu skasuj i pliki i katalogi
99. Skopiuj wszystkie pliki z katalogu /etc to katalogu ~/etc (bez podkatalogów), a następnie wyświetl je i następnie usuń wszystkie pliki z rozszerzeniem conf. Nie zwracaj uwagi na komunikaty o błędach, a będzie ich sporo
100. Utwórz spakowane archiwum wszystkich plików conf z katalogu /etc. Archiwum wynikowe ma mieć nazwę /etc_conf.tgz. Do utworzenia archiwum wykorzystaj polecenia tar i gzip Do tego zadania hintera nie będzie
101. Utwórz 10 plików o nazwach od 0 do 9. Następnie utwórz plik o nazwie wszystkich plików i katalogów z katalogu bieżącego. Wyświetl zawartość katalogu. Skasuj wszystkie utworzone pliki
102. A jakbyś zrobił to samo zadanie, gdyby zamiast 10 plików trzeba byłoby utworzyć takich plików o 0 do 99 lub więcej
103. Z pewnością na swoim dysku Z masz jakieś pliki java i pliki tekstowe. Wyświetl ile linii ma każdy z nich. Zrób to tylko dla wskazanych przez siebie katalogów
104. Ze strony https://www.lipsum.com/feed/html ściągnij 50 paragrafów tekstu Lorem ipsum i zapisz je w pliku tekstowym. Usuń ręcznie puste linie (Jak potrafisz to możesz usunąć automatycznie) i policz te które pozostały. Policz również i znaki i słowa. Na dole strony mamy krótki raport z liczby linii, słów i znaków. Czy polecenie wc dokładnie tak policzyło jak na stronie? Jeżeli nie, wytłumacz dlaczego nie.
105. Wyświetl wszystkie katalogi z podkatalogami z katalogu /bin i /usr. Zmierz ile czasu zajęła ta operacja.
106. To zadanie jest trochę bardziej skomplikowane i wymagać może przejrzenia literatury:\nUruchom polecenie (pewnie domyślasz się co się stanie): asmyk@msh:~\$ sleep 1\n A teraz: asmyk@msh:~\$ time sleep 1\n A teraz

uruchommy dwa polecenia sleep i zmierzmy czas ich wykonania: asmyk@msh:~\$ time { sleep 1 ; sleep 1 ; } \n I tutaj jest zadanie: zmieńmy teraz pierwszy średnik w ampersand: asmyk@msh:~\$ time { sleep 1 & sleep 1 ; } \n Potrafisz to wytłumaczyć?

07. Do katalogu /tmp/numerIndeksu skopiuj 4 najmniejsze pliki z katalogu /usr/bin, które nie są łączami symbolicznymi, a następnie zmień ich nazwy odpowiednio na p1, p2, p3 i p4
08. Jak byś skopiował jeden plik do dwóch różnych lokalizacji. Ja mam jakiś pomysł, ale może masz lepszy, chętnie go poznam.
09. Wyświetl nazwy wszystkich plików tekstowych (*.txt) i skryptów (*.sh) z 6 wybranych katalogów
10. Zamiast nazw plików, wyświetl zawartość tych plików
11. We wskazanym przez użytkownika katalogu, znajdź wszystkie pliki niewykonywalne ale takie które możemy odczytać, są odczytywalne dla człowieka i wyświetl ich nazwy i zawartość, ale w zawartości zamień wszystkie duże litery na małe, a małe na duże.
12. Dodatkowo usuń z wyniku wszystkie cyfry parzyste
13. We wskazanym przez użytkownika katalogu, znajdź wszystkie pliki Wykonywalne i uruchom je
14. Zmodyfikuj rozwiązanie, aby obsługiwało ono więcej katalogów, które będą przeszukiwane. Aktualnie przeszukiwany jest tylko jeden katalog
15. Zmodyfikuj rozwiązanie, aby obsługiwało ono tyko skrypty i żeby nie były przeszukiwane podkatalogi - użyj polecenia man find.
16. Wyświetl bieżący rok na konsoli. Tylko rok
17. Napisz wyrażenie, które wyliczy ilość sekund od daty 1970-01-01 do początku dnia dzisiejszego. Załóżmy dla ułatwienia, że każdy miesiąc ma 30 dni.
18. Przedstaw ciąg operacji, wyznaczający liczbę sekund między aktualna godziną, a godziną o której rozpocząłeś dzisiaj pracę
19. Utwórz 10 plików o nazwach od 0 do 9. Następnie utwórz plik o nazwie wszystkich plików i katalogów z katalogu bieżącego. Wyświetl zawartość katalogu. Skasuj wszystkie utworzone pliki.
20. Powtórz to samo zadanie, ale zamiast plików wykonaj operacje na katalogach
21. Znajdz wszystkie pliki z rozszerzeniem java i txt w swoim katalogu domowym wraz z podkatalogami i policz ile każdy z nich ma linii
22. Obsłuż pliki, których nazwa zaczyna się na litery A, B lub C. Resztę plików pomiń
23. Użytkownik z klawiatury podaje nazwę katalogu, który chcemy skasować. Jeżeli nam się to udało powinniśmy wyświetlić komunikat OK, a jak nie to komunikat NOK

24. Tak zmodyfikuj ten program, aby usuwał plik, którego nazwa jest podana z klawiatury. Dodatkowo słowa OK lub NOK mają być wyświetlone z użyciem plecenia banner
25. Napisz wyrażenie, które policzy ilość sekund od daty 0000-00-00 00:00:00 do początku dnia dzisiejszego. Załóżmy dla ułatwienia, że każdy miesiąc ma 30 dni
26. Jak zmieniłoby sie rozwiązanie tego zadania, jeżeli przyjelibyśmy, że średnio każdy miesiąc ma 30,41666667 dni. Użyj kalkulatora bc
27. Napisz ciąg poleceń, które spowodują zamianę pierwszej połowy linii z wskazanego pliku z drugą połową linii tegoż pliku. Nie zwracamy uwagi na to czy plik ma parzystą liczbę linii czy nie.
28. Rozwiąż to samo zadanie, ale zamień pierwszą jedną trzecią linii z ostatnią jedną trzecią linii. W jednej trzeciej linii, która nie była zamieniana, zamień linie kolejnością. Przydatne może być polecenie tac
29. Napisz skrpt, który wyświetli ile mamy ścieżek w zmienne systemowej PATH
30. Napisz skrypt, który pobierze od użytkownika ścieżkę, a nasz skrypt doda to do zmiennej systemowej PATH i wyświetli tak zmodyfikowaną zmienną PATH
31. Ta praca domowa, jest dokładnie taka sama jak #30031_1, ale z tą różnicą, że zmienna PATH ma istotnie zmieniać swoją wartość w systemie. Uwaga, nie tylko ma być ta zmiana widoczna w czasie działania skryptu, ale również po jego zakończeniu
32. Napisz skrypt, który pobierze z klawiatury tekst w formacie 0:1:1:2:3:5:8:13:21:34. Jak widać jest to ciąg Fibonacciego. Skrypt ma wyznaczyć kolejną liczbę Fibonacciego i dokleić ją na końcu do otrzymanego ciągu (po dwukropku), a wynik wyświetlić na konsoli. Zakładając, że skrypt nazywa się f.sh, chciałbym go uruchomić w następujący sposób: echo \"0:1\" | ./f.sh | ./f.sh | ./f.sh | ./f.sh ...
33. Zmień nasze rozwiązanie, aby zamiast ciągu Fibonacciego generowany był ciąg arytmetyczny.
34. Zmień nasze rozwiązanie, aby zamiast ciągu Fibonacciego generowane były kolejne silnie
35. Napisz skrypt, który pobierze dwa argumenty, pierwszym będzie nazwa pakietu języka JAVA, a drugim nazwa klasy i na ich podstawie w odpowiednim miejscu na dysku utworzy prawidłową klasą języka JAVA z odpowiednią klasa publiczną i funkcją main wyświetlająca tekst Hello world.
36. Dodaj do naszego generatora możliwość generowania komentarza, kto, kiedy i w jakim celu ten program wygenerował
37. Jest to kontynuacja zadania 3004. Umiemy już napisać prosty skrypt, który utworzy plik java w odpowiednim pakiecie. Dodajmy teraz funkcjonalność warunkowego

budowania elementów klasy: może ona zawierać (lub nie) funkcję main i może zawierać (lub nie) kod wyświetlający tekst Hello world

38. Wzbogać nasz generator o możliwość generowania pól (powiedzmy dla kilku podstawowych typów prostych), getterów i setterów.
39. Wzbogać nasz generator o możliwość generowania metody toString.\nHomework
40. Wzbogać nasz generator o możliwość generowania kodu wywołującego konstruktor i metodę toString
41. Pamiętasz zadanie 3002? Jeżeli nie to cofnij się do niego. Jak już je rozwiązałeś i rozumiesz tamto zadanie, to teraz obsłuż przynajmniej dwa błędy: otrzymany ciąg musi mieć przynajmniej dwa elementy i muszą to być liczby. Powinniśmy jeszcze sprawdzić mnóstwo innych rzeczy, ale na razie to wystarczy
42. Jeżeli znasz już pętle to dopisz kod sprawdzający, czy otrzymany ciąg liczb jest naprawdę ciągiem Fibonacciego
43. Jest to kontynuacja zadania 3005. Uzupełnij powstały skrypt o dodatkową funkcjonalność, a mianowicie skrypt może pobrać dodatkowy argument -s (--silent), który będzie przełączał skrypt w tryb cichy to znaczy nie będa raportowane żadne komunikaty na konsolę
44. Dodaj funkcjonalność polegającą na dodaniu obsługi trybu log, czyli komunikaty będą raportowane i na konsolę (jeżeli nie mamy trybu silent) i do pliku.
45. Jak wykonałeś poprzednią pracę domową, musiałeś jakoś określić nazwę pliku. Teraz ja chcę, aby nazwa pliku zawierała datę, godzinę i minutę. Każdy zapis musi być zaraportowany tak aby logi a danej minuty pojawiały się w konkretnym pliku. Jak się zmieni minuta to kolejne logi zostaną utworzone już w kolejnym pliku. Dodatkowo, każdy log musi mieć informację o dokładnym czasie powstania logu (sekundy, milisekundy), i jaki użytkownik go wygenerował oraz na jakiej maszynie to się stało - adres ip może by się przydał.
46. Jest to kontynuacja zadania 3006. Mamy już generator klas. Jeżeli zrobiliście prace domowe, to macie logowanie, a jak nie to nie macie. Teraz czas aby coś zrobić z tymi utworzonymi klasami. Napiszmy walidator, który wstępnie zweryfikuje czy klasy są poprawne. Co będziemy sprawdzać? Czy klasy są w odpowiednich pakietach i czy są klasa publiczna zapisana w pliku z rozszerzeniem java ma odpowiednią nazwę.
47. Zapewne znasz polecenie mv - służy ono do zmiany nazwy plików lub przenoszenia ich między katalogami. Zaimplementuj własną wersję tego polecenia. Ma ono oczywiście dotyczyć naszych plików java, a jego zadaniem ma być zarówno przeniesienie pliku z katalogu do katalogu, jak również zmiana nazwy pakietu wewnątrz pliku

48. Zmień to rozwiązanie na podprogramy. To już robiłeś, ale przy tej liczbie operacji, aż się prosi o taki sensowny, przemyślany podział
49. W powyższym rozwiązaniu założyłem, że przenosimy plik java do innego pakietu, czyli do innego katalogu, ale nie zmieniamy samej nazwy tego pliku. Wprowadź zmianę w skrypcie aby obsługiwał on również przypadek przeniesienia klasy z pakietu do pakietu z jednoczesną zmianą nazwy. Uwaga, oprócz zmiany nazwy pliku, musisz zmienić nazwę klasy publicznej w tym pliku. Obsłuż tylko najprostsze przypadki
50. W zadaniu 3008 napisaliśmy skrypt, który w zalezności od sytuacji kończył się różnymi kodami błędów. Błędy te w żaden sposób nie są wyjaśnione w sposób przyjazny dla użytkownika. Napisz skrypt, który będzie wyświetlał odpowiednią informację tekstową, w zależności od uzyskanego kodu błędu
51. Zaproponuj własne rozwiązanie do obsługi kilku języków. W zależności od ustawień komunikat może być wyświetlony po angielsku, polsku, hiszpańsku itd...
52. Utwórz, strukturę katalogów zgodną ciągiem liczb Fibonacciego, czyli w bieżącym katalogu tworzymy jeden katalog, w tym katalogu tworzymy też jeden katalog, w tym katalogu, tworzymy dwa katalogi, w tych dwóch tworzymy trzy katalogi, a w każdym z tych sześciu utworzonych tworzymy pięć katalogów. Utwórz to dla dowolnej nieujemnej wprowadzonej z klawiatury liczby całkowitej, ale powiedzmy nie większej niż 10. Zakładamy, że wprowadzona dana będzie liczbą
53. Napisz podprogram, który zwróci informację, ile takich katalogów zostanie fizycznie utworzonych.
54. Napisz skrypt, który wczyta z klawiatury liczbę całkowitą i tekstowo na konsoli "narysuje" kopertę
55. Zmodyfikuj napisany skrypt, aby obsługiwał też koperty o kształcie prostokąta.
56. Zmodyfikuj napisany skrypt, aby sprawdzał wartość zwracaną przez return, a nie to co jest przez co ten podprogram wyświetlane (usuń wywołania typu echo 'T' lub echo 'F' z podprogramu.
57. Zamień podprogram is_printed_part_of_envelop na print_pixel_of_envelope. Podprogram print_pixel_of_envelope ma drukować konkretny piksel, czyli albo X albo kropkę.
58. Napisz skrypt drukujący choinkę o podanej wysokości
59. Napisz skrypt, który wyznaczy statystykę (histogram) rozmiaru plików dla wskazanego katalogu. Skrypt przyjmuje informacje o klasie rozmiaru pliku w postaci: 0-100,200-2000,3000-4000. W tym przypadku mamy trzy klasy. Do pierwszej klasy należą pliki małe od 0B do 100B, do kolejnej od 200B do 2000B i do ostatniej pliki od 3000B do 4000B
60. Rozszerz działanie naszego skryptu, aby przetwarzał wiele katalogów.\nHomework

31. Dodaj informację ile czasu zajęło skryptowi wykonanie tej operacji, ile czasu średnio przetwarzał jeden plik. Zbadaj, który z podprogramów, która operacja zajmuje najwięcej czasu. Może da się to jakoś zoptymalizować, przepisać
32. Przenieś interwały z ciała skrytpu do pliku konfiguracyjnego i odczytaj je przed rozpoczęciem tworzenia histogramu
33. Dodaj informacje, ile procentowo zajmują pliki w każdym interwale
34. Teraz trochę inne zadanie, a dokładniej mówiąc dwa zadania. Napisz dwa skrypty. Pierwszy z nich, nazwijmy go wifi-collector, będzie zbierał (imitował zbieranie) danych o natężeniu sygnału sieci wifi w danym miejscu. Dane te zapisze do pliku i operację tę będzie okresowo powtarzał . Dane z tego pliku będzie pobierał wifi-presenter i przedstawiał je na tekstowym wykresie słupkowym. Oba skrypty mają działać w nieskończoność. Jak wifi-collector nie udostępnił danych to wifi-presenter ma czekać na dane. Format przekazywanych danych oraz sposób ich prezentacji pokazałem poniżej
35. Rozszerz działanie wi-fi presentera tak aby na końcu każdego wskaźnika była wyświetlona wartość procentowa względem wartości 80. Czyli jak wskaźnik wyświetla wartość 39 to powinniśmy wyświetlić się tekst 48,75%%
36. Zmień działanie wifi-presentera tak aby sam wykres wyświetla się nie od lewej strony, ale od prawej.
37. Jak kilka osób w tym samym czasie, będzie uruchamiać te skrypty, może nie działać to poprawnie. Dlaczego? Wszyscy korzystają z tych samych plików w tym samym czasie. Popraw to aby takich kolizji uniknąć
38. Wprowadzamy zmianę w wifi-collectorze. Oprócz siły sygnału, udostępnia on dodatkowe informacje: numer kanału danej sieci wifi, oraz aktualną maksymalną przepustowością. Zmodyfikuj nasz plik oraz wprowadź nowy sposób wizualizacji - kolejny wykres, na którym przedstawisz przepustowość. Sposób przedstawienia numeru kanału pozostawiam Tobie
39. Będzie to bardzo praktyczne, ale i trochę trudne zadanie. Zajmiemy się badaniem wypadków morskich. Niestety zdarzają się one i jeden z nich miał miejsce 19 lipca 2020 w porcie w Gdyni. Holownik Uran uderzył w nabrzeże portowe. Dokładny raport z tego wypadku możecie Państwo znaleźć https://pkbwm.gov.pl/wp-content/uploads/2021/07/Raport-uproszczony-050_20-Uran.pdf. Każda jednostka pływająca wyposażona jest (powinna być) w urządzenie GNSS, raportujące położenie w określonym czasie. Takie urządzenie jest też na holowniku Uran. Przed kolizją zbierało ono współrzędnej położenia statku i czas kiedy pomiar został wykonany. Na potrzeby tego raportu został napisany program w języku Pyton, który na podstawie zebranych współrzędnych i czasów, wyznaczył drogę przebytą przed kolizją i prędkość z jaką poruszał się holownik. Przepisz ten program na basha. Nic nie zmieniaj, po prostu przepisz i przetestuj