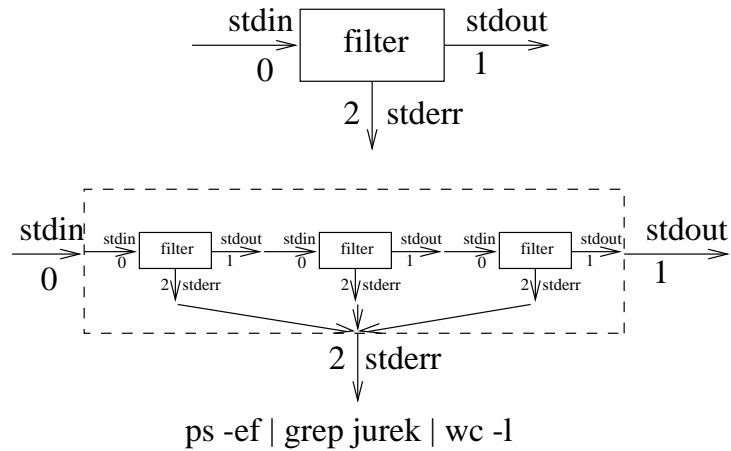


Filtry



1

Grupowy opis plików

- * dowolny ciąg znaków
- ? dowolny znak
- [...] dowolny znak z klasy
- [a-z] dowolny znak pomiędzy a i z
- [!...] dowolny znak spoza klasy

3

Kierowanie wejścia/wyjścia

- > – skierowanie wyjścia
- 2> – skierowanie błędów
- < – kierowanie wejścia
- >> – skierowanie wyjścia (dopisywanie)
- | – skierowanie wyjścia jednego programu na wejście drugiego
- 2>&1 – skierowanie błędów na standardowe wyjście

2

Zmienne

- PATH** – ścieżka poszukiwań komend
- HOME** – katalog użytkownika
- MAIL** – skrzynka pocztowa użytkownika
- SHELL** – nazwa powłoki
- PS1** – znak zachęty
- PS2** – znak zachęty (używany przy kontynuowaniu komendy)

4

Argumenty

- `$0` – nazwa komendy
- `$1` – pierwszy argument
- `$i` – i-ty argument
- `$*` – wszystkie argumenty z wyjątkiem `$0` jako jeden łańcuch znaków
- `$@` – wszystkie argumenty z wyjątkiem `$0` jako osobne łańcuchy znaków
- `$#` – liczba argumentów
- `$?` – status wyjścia ostatniej komendy
- `$$` – PID aktualnej powłoki
- `$!` – PID ostatniej komendy wykonywanej w tle

5

Znaki specjalne

Znaki specjalne: ; & () | ^ < > nowa_linia odstęp tabulator

- `\c` – znak *c*
- `'string'` – chroni wszystkie znaki specjalne
- `"string"` – chroni wszystkie znaki specjalne z wyjątkiem `$`
- `'command'` – podstawienie rezultatu komendy

7

Podstawienia

- `${name}` – wartość parametru
- `${name:-word}` – jeśli parametr nie jest zdefiniowany podstaw słowo
- `${name:=word}` – jeśli parametr nie jest zdefiniowany zdefiniuj go
- `${name:?word}` – jeśli parametr nie jest zdefiniowany wypisz słowo i zakończ działanie
- `${name:+word}` – jeśli parametr jest zdefiniowany podstaw słowo

6

Komendy wewnętrzne

- `break` – wyjście z pętli
- `continue` – skok na początek pętli
- `cd` – zmiana katalogu
- `echo` – wyświetlenie tekstu
- `export` – udostępnienie zmiennych innym programom
- `read` – wczytanie jednej linii z wejścia
- `test` – sprawdzenie warunku
- `unset` – skasowanie definicji zmiennej

8

Sterowanie

- **for** *zmienna* [*in słowo ...*] **do lista done**
– powtórzenie listy komend dla wszystkich wymienionych wartości zmiennej
- **case** *słowo in* [*wzorzec* [| *wzorzec*] ...) *lista* ; ;] ... **esac**
– wykonanie listy komend jeśli słowo pasuje do wzorca
- **if** *wyrażenie then lista* [**elif** *lista then lista*] ... [**else lista**] **fi**
– wykonanie listy komend zależnie od wartości wyrażenia
- **while** *wyrażenie do lista done* – powtarzanie listy komend dopuki wyrażenie jest prawdą
- *(lista)* – wykonanie listy komend w osobnej powłoce
- { *lista*; } – wykonanie listy komend w tej samej powłoce
- *nazwa()*{ *lista*; } – zdefiniowanie procedury
- **#komentarz** – komentarz

9

Porównania

- z **łańcuch** – Prawda jeśli długość łańcucha jest równa zero
- n **łańcuch** – Prawda jeśli długość łańcucha jest większa niż zero
- s1 = s2** – Prawda jeśli łańcuchy są identyczne
- s1 != s2** – Prawda jeśli łańcuchy nie są identyczne
- n1 -eq n2** – Prawda jeśli liczby są równe
- n1 -ne n2** – Prawda jeśli liczby są różne
- n1 -lt n2** – Prawda jeśli n1 jest mniejsze niż n2
- n1 -le n2** – Prawda jeśli n1 jest mniejsze lub równe n2
- n1 -gt n2** – Prawda jeśli n1 jest większe niż n2
- n1 -ge n2** – Prawda jeśli n1 jest większe lub równe n2

11

Sprawdzenie plików

- r **plik** – Prawda jeśli plik istnieje i można go odczytać
- w **plik** – Prawda jeśli plik istnieje i można go zapisać
- x **plik** – Prawda jeśli plik istnieje i można go wykonać
- f **plik** – Prawda jeśli plik istnieje i jest zwykłym plikiem
- d **plik** – Prawda jeśli plik istnieje i jest katalogiem
- h **plik** – Prawda jeśli plik istnieje i jest linkiem symbolicznym
- s **plik** – Prawda jeśli plik istnieje i ma rozmiar większy niż zero

10

```
# Czekanie na odpowiedz t lub n
getyn()
{
    while echo "\n$* (t/n)? \c">&2
    do read yn rest
        case $yn in
            [tT]) return 0 ;;
            [nN]) return 1 ;;
            *) echo "Odpowiedz t lub n" >&2 ;;
        esac
    done
}
getyn "Czy chcesz zachować plik"
if [ $? -eq 0 ]; then
....
```

12