

Języki skryptowe

- Programy są na bieżąco tłumaczone i wykonywane przez interpreter
- Programy w językach kompilowanych są najpierw tłumaczone na język maszynowy, potem wykonywane
- Wolniejsze

Skrypty powłoki

- Prostota języka (krótsze)
- Przenośność (POSIX)
- Łatwość programowania (dużo użytecznych narzędzi)

```
$ cat > nusers
```

```
#!/bin/sh
```

```
who | wc -l
```

```
^D
```

```
$ chmod +x nusers
```

```
$ ./nusers
```

Etapy uruchomienia:

1. Utworzenie powłoki potomnej
2. Przeszukanie ścieżki
3. Wczytanie kodu programu i zastąpienie nim kodu powłoki
4. Zainicjowanie wykonania programu, oczekiwanie na jego zakończenie

Zmienne

- `jmr=jan`
- `echo $jmr`
- `jmr=jan maria rokita (ŹLE)`
- `jmr=„jan maria rokita”`
- `printf „$jmr \t $jmr %d\n” 25`
- `PATH=$PATH:.` (odradzane, ułatwia atak)

Standardowe we/wy

- Zmiana standardowego wejścia <
- Zmiana standardowego wyjścia > >>
- Potoki | (bezpośrednio, zamiast plików)
- set | grep PATH
- tr 'Litwo' 'Ojczy' < x.txt | sort > x2.txt
 - Ile Cje crzeba cenjc
 - Ojczy Ojczyzny myja
 - Ty jesces jak zdryzje

Argumenty

- finduser
 - who | grep \$1
- ./finduser marcin
- ./finduser (bez argumentu)
- Śledzenie: sh -x finduser marcin:
 - + who
 - + grep marcin
- set +x – wyłączenie śledzenia:
 - + set +x

Wyrażenia regularne

- Notacja wzorców pozwalająca na dopasowanie całych klas ciągów w trakcie przeszukiwania tekstu. Np. wszystkie słowa na literę „a”.
- BRE (Basic Regular Expressions) – używane przez grep, sed, vi
- ERE (Extended Regular Expressions) – używane przez egrep (grep -E), awk, lex
- fgrep (lub grep -F) – fast grep, zwykłe ciągi znaków
- Używane przez tr, sed, awk, Perl, Python, C itp.

Metaznaki

- \ - wyłącza znaczenie specjalne
- . - dowolny pojedynczy znak
- * - 0 lub więcej poprzedzających pojedynczych znaków
- ^ - kotwica początku
- \$ - kotwica końca
- [...] - wyrażenie nawiasowe, dopasowuje dowolny znak wymieniony w nawiasach, - w środku oznacza przedział, ^ to negacja

Metaznaki

- $\{n,m\}$ - (tylko BRE) wyrażenie przedziałowe; od n do m wystąpień poprzedzającego znaku
- $\{n,m\}$ - (tylko ERE) wyrażenie przedziałowe; od n do m wystąpień poprzedzającego znaku
- $\backslash(\backslash)$ - (tylko BRE) buforowanie podwzorca
- $\backslash n$ - (tylko BRE) użycie n -tego podwzorca
- $+$ - (tylko ERE) 1 lub więcej wystąpień poprzedzającego wyrażenia
- $?$ - (tylko ERE) 0 lub 1 wystąpień poprzedzającego wyrażenia

Metaznaki

- | - (tylko ERE) alternatywa: wyrażenie przed lub po
- () - zamknięta grupa wyrażeń regularnych

Klasy znaków

- `[:alnum:]` - znaki alfanumeryczne (litery i cyfry)
- `[:alpha:]` - litery
- `[:blank:]` - spacja i tabulator
- `[:cntrl:]` - znaki sterujące
(np. `tr -d '[:cntrl:]' < x.txt` usunie wszystkie znaki sterujące)
- `[:digit:]` - cyfry
- `[:graph:]` - znaki o niepustej reprezentacji graficznej

Klasy znaków

- [:lower:] - małe litery
- [:print:] - znaki drukowalne
- [:punct:] - znaki interpunkcyjne
- [:space:] - znaki odstępow
- [:upper:] - wielkie litery
- [:xdigit:] - cyfry szesnastkowe

- Litwo - ciąg „Litwo”
- Kotwice: ^ i \$
- `grep „^Litwo” < x.txt` - wystąpi
- `grep „Litwo$” < x.txt` - nie wystąpi
- `grep „^Litwo$” < x.txt` - nie wystąpi (wiersz składający się ze słowa „Litwo”)
- Litw[oaie]
 - Litwo
 - Litwa
 - Litwi (także Litwie)
 - Litwe

- „Litw.” - wszystko
- „Litw. „ - bez „Litwie”
- „Litw.* ” - wszystko
- „Litw[a-z] „ - bez „Litwie”
- „Litw[a-z][a-z] „ - tylko „Litwie”
- „Litw[a-z]* „ - wszystko
- „Litw[a-z]+ „ - wszystko (tylko ERE)
- „Litw[^0-9]{1,1} „ - bez „Litwie” (BRE)
- „Litw[^0-9]{1,1} „ - bez „Litwie” (ERE)
- „\ (Litw[a-z] \)\1” - „Litwo Litwo” lub „Litwa Litwa”, ale nie „Litwo Litwa”; pomocne dla znajdowania par cudzysłówów i apostrofów

- „Litw[^o]”
 - „Litwo Ojczyzna moja” NIE
 - „Litwo Litwa” TAK
- „^Litw[a-z]? „ - bez „Litwie” (tylko ERE)
- „^Litwa[[]],” - „Litwa]”
- „^Litw[-a] „ - „Litwa ” lub „Litw- ”
- (Litwa|Litwo) - „Litwa” lub „Litwo” (tylko ERE)
- „^\$” - pusty wiersz

Zadania

- Zastąpić wszystkie litery w tekście literą „z”
 - `tr '[:alpha:]' 'z'`
- Znaleźć ciągi „ala[n spacji]ma[m spacji]kota”
 - `grep -E „ala [[:space:]]*ma [[:space:]]*kota”`