SYSTEMS LABORATORY

Cursul #6

Automatizarea sarcinilor. Shell Scripting

Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems.

Jamie Zawinski (JWZ)

Suport de curs

Capitolul 13 - Automatizarea sarcinilor

- https://github.com/systems-cs-pub-ro/carte-uso/releases

Ce este un Script Shell

- O înșiruire de comenzi shell
- Este interpretat de shell: se interpretează pe rând fiecare comandă
- Pe lângă comenzi externe, sunt folosite și comenzi interne, precum **if, for, while, case**
- Facilități shell precum *command expansion*

Perspective pentru shell scripting

- quick'n'dirty: dacă e prea complicat trebuie apelat la un limbaj de programare
- folosește comenzi și resurse existente
  - nu reinventa roata
- există mai multe moduri de a face un lucru
  - alege cel mai bun pentru situația dată
- comunicare folosind text
- atenție la caractere speciale
- niciodată nu vei ști suficient de mult shell scripting: documentează-te, exersează și îmbunătățește

Rulare script

- bash myscript.sh
- chmod a+x myscript.sh && ./myscript.sh

- #! -> shebang
  - indică interpretorul scriptului

Exemplu Shell Script

```
#!/bin/bash

sudo mdutil -a -i off
sudo launchctl unload -w
/System/Library/LaunchDaemons/com.apple.metadata.mds.plist
sudo launchctl load -w
/System/Library/LaunchDaemons/com.apple.metadata.mds.plist
sudo mdutil -a -i on
sudo mdutil -E /
```

Utilizarea Shell Scripting

- automatizare
  - înlănțuire comenzi pentru un rezultat mai complex
  - task-uri repetitive
  - se rulează scriptul la nevoie
- prelucrare de date
  - task-uri simple
  - quick’n’dirty
  - se folosesc filtre de text: prelucrează text (head, tail, grep, cut, tr, awk, sed)

One liner

- Înlănțuirea unor comenzi
- De obicei legate prin operatorul pipe (|)
- Ceva util _acum_, nu neapărat permanent, in schimb un shell script poate fi refolosit
- Orice shell script poate fi un one liner

- http://www.bashoneliners.com

Dezvoltarea de scripturi shell

Comenzi existente

- Nu trebuie să fie complicat
  - acolo unde o comandă are opțiuni utile, se folosesc acelea
- Comenzi de afișare: NU se folosesc în scripturi
  - ls, ps, df, ifconfig, w
- Comenzi de prelucrare: se folosesc în scripturi
  - stat, find, fișierele din /proc, grep

Înlănțuire de comenzi

- pipe (operatorul |)
- expandarea comenzilor: $(...)
- folosire nouă a comenzilor existente
- preluarea outputului unei comenzi
  - la intrarea standard a alteia (pipe)
  - ca argument al alteia (expandare)

Variabile

- stocare de valori
- nu au un tip de date
- $?: codul de ieșire al ultimei comenzi
- $$: PID-ul procesului curent
- $#: numărul de parametri în linia de comandă
- $1, $2, ...: al N-lea parametru

Controlul fluxului

- instrucțiuni de decizie: if, case
- intrucțiuni de ciclare: for, while

Construcția if

- De obicei, if comanda1 e de forma:
  - If test conditie

```
if comanda1; then
        comanda2
else
        comanda3
fi
```

- Verifică dacă există fisierul:

```
if test –f ”$file”; then
   echo ”File $file exists.”
fi
```

- Verifică dacă există argumente:

```
if test ”$#” –ne 1; then
   echo ”Usage: $0 argument”
   exit 1
fi
```

Construcția for

- List e o construcție asemănătoare cu o listă

```
for i in list; do
      command1
      command2 ...
done
```

- Parcurgerea utilizatorilor din sistem

```
for user in $(cut –d ‘:’ –f 1); do
    nlogins=$(last ”$user” | grep ”^$user” | wc -l)
    echo ”User $user logged in $nlogins times recently.”
done
```

Construcția while read

- parsing cu while read
- mai puternic decât cut, mai slab decât awk
- IFS: Input field separator
- Extragerea numelui de utilizator și a directorului home

```
while read field1 field2 ...; do
    command1
    command2 ...
done < file
```

```
IFS=":"; while read user x y z t home v; do echo "$user:$home";
        done < /etc/passwd
```

For vs while

- for pentru elemente dintr-o listă
  - listă de utilizatori
  - listă de fișiere
- while pentru parsing (împărțire elemente de pe o linie în bucăți)
  - date tabelare pentru prelucrare

Funcții

- mai rar folosite ca într-un limbaj de programare
- înglobarea unor acțiuni
- se apelează similar unui script
- pot avea argumente:
  - $1, $2 ... devin argumentele funcțiilor

Exemplu funcții

```
#!/bin/bash
OFFLINEIMAP=/opt/local/bin/offlineimap
LOGFILE=~/.log/offlineimap-cron.log

function check_alive()
{
        ps -ef | grep "$OFFLINEIMAP" | grep -v grep > /dev/null 2>&1
}

check_alive
if test $? -ne 0; then
        nohup nice -n 19 "$OFFLINEIMAP" >> "$LOGFILE" 2>&1 &
fi

exit 0
```

Expresii regulate - Utilizare

- Regular expressions sau regex
- Cautare (searching, pattern matching)
- Validarea unui șir, a unui text, a unei intrări (număr de telefon, URL, nume de variabilă)
- Substituirea unei expresii

Când să nu folosești expresii regulate

- Când existe parsere
- Când există aplicații, tool-uri, funcții mai bune pentru a face aceste lucru
- Pentru anumite tipuri de date (adrese de e-mail, HTML): fie nu se poate, fie e foarte greu
- Atunci când folosite abuziv, fac codul nementenabil/nelizibil

Metacaractere în expresii regulate

- **^ :** început de linie
- **$:** sfârșit de linie
- **. :** orice caracter
- **[...] :** set de caractere
- **? :** expresia anterioară cel mult o dată
- *** :** expresia anterioară de oricâte ori posibil niciodată
- **+ :** expresia anterioară de oricâte ori cel puțin o dată
- **e1|e2 :** expresia de dinainte sau cea de după

Exemple de expresii regulate

- **[_a-zA-z][_0-9a-zA-Z]***
  - Nume de variabilă/funcție
- **07[:digit:]{8}**
  - Număr de telefon
- **[:upper:][:alpha:]+**
  - Numele unei persoane
- **^[:upper:]{1,10}$**
  - Linii conținând maxim 10 majuscule
- **[:digit:]{2}\.{pdf|png|svg}**
  - Nume fișiere din două cifre și diferite extensii

Regex vs globbing

- Globbing este folosit în shell, în special pentru filename expansion
- Globbing poate fi considerat o formă mai slabă de expresie regulate
- Expresiile regulate sunt puternice cu mai multe cazuri de utilizare și suport în majoritatea limbajelor/framework-urilor moderne

Suport expresii regulate

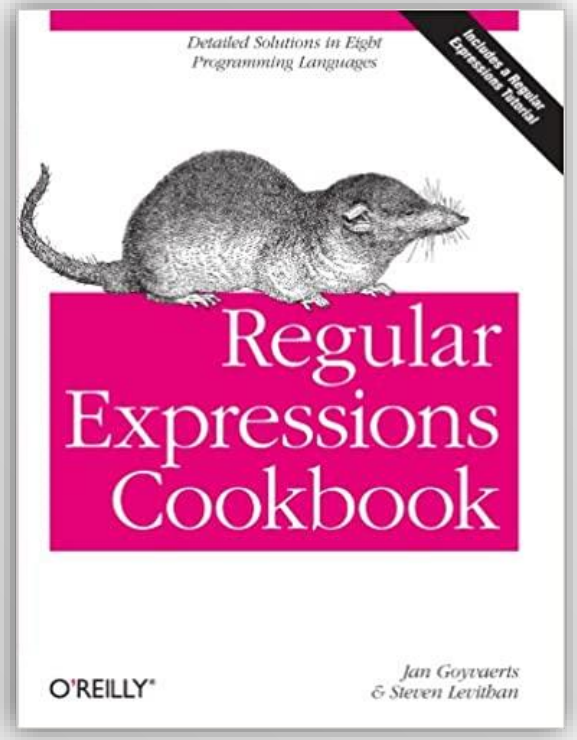
- **Utilitare shell:** grep, awk, sed
- **Limbaje de programare:** Perl, Python, PHP, Ruby, Java, JavaScript, C++11
- **Editoare:** Vim, Emacs
- **Baze de date**
- **Biblioteci cu suport de expresii regulate:** Gnulib

Sfaturi Shell Scripting

- nu folosiți shell scripting pentru ceea ce se face mai bine/ușor/eficient în Python, C, Java, Ruby, PHP etc
- nu reinventați roata
- folosiți ghilimelele când referiți valoarea unei variabile
- folosiți cea mai bună opțiune (cut, grep, tr, awk, sed, while read) după nevoie
- folosiți opțiuni de tip expresii regulate sau similare între apostrofuri

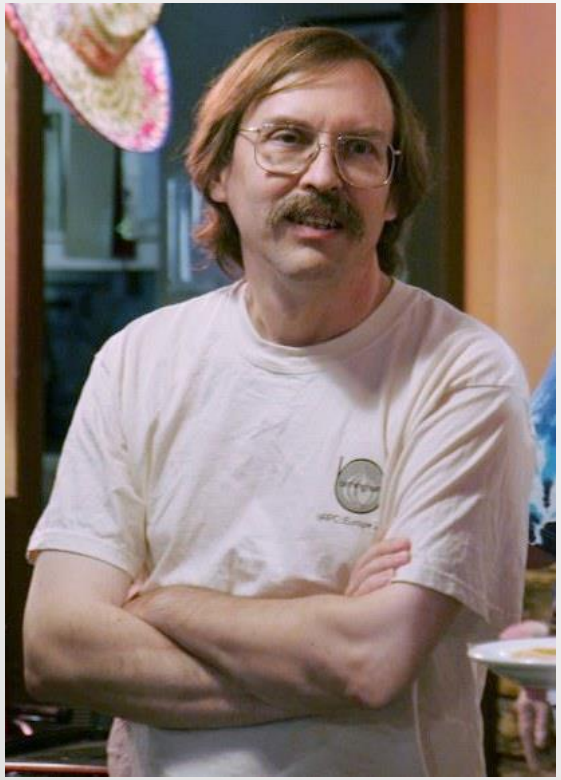
Regular Expressions Cookbook

- Jan Goyvaerts, Stevemn Levithan
- creatorii http://www.regular-expressions.info/, http://regexpal.com/
- 2nd Edition
- practică, multe „rețete”
- o recenzie aici: https://blog.codinghorror.com/regular-expressions-for-regular-programmers/



Larry Wall

- creatorul Perl
- BDFL pentru proiectorul PERL
- autorul programului patch
- câștigător al *International Obfuscated C Code Contest*



Lua

- limbaj de programare de scripting
- paradigme multiple (la fel ca Python)
- proiectat pentru a fi ușor încorporat în alte limbaje
- API C simplu
- limbaj de scripting pentru dezvoltatorii de jocuri
- folosit de limbajul de scripting al utilitarului nmap
- folosit pentru configurarea managerului de ferestre Awesome

Cuvinte cheie

- variabile
- variabile de mediu
- one line
- filtru de test
- script shell
- IFS
- while read
- for
- if
- expresii regulate
- metacaractere