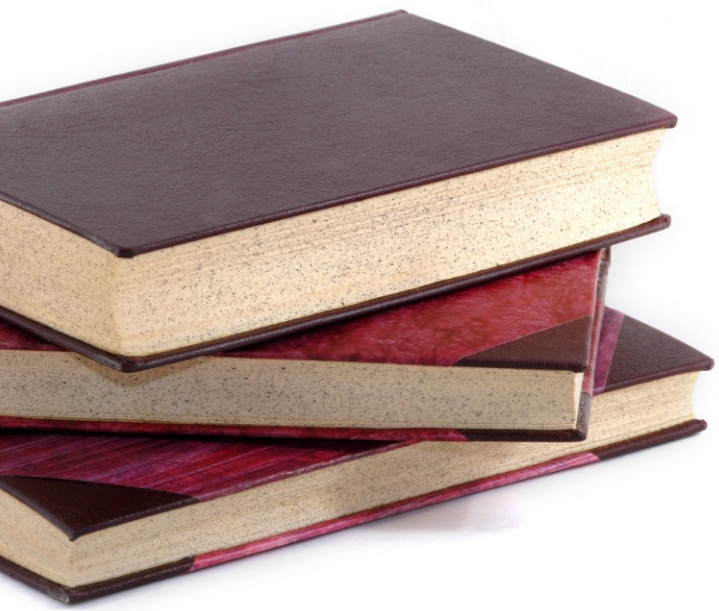
just crunch it

Cursul 12



12

Shell scripting

13, 19 decembrie 2011

Moto

Back off, or I'll replace you with a small shell script.

Securitate

privilegii

resurse

defense in depth

weakest link

riscuri

parole

acces

educație

phishing

monitorizare

mentenanță

updates

configurare

limitări

rootkit

sudo

root

drepturi de acces

umask

ACL

servicii

criptare

firewall

Security question

Qualifying question

Just to prove you are a human, please answer the following math challenge.

Q: Calculate:

$$\frac{\partial}{\partial x}\left[4 \cdot \sin\left(7 \cdot x - \frac{\pi}{2}\right)\right]\Big|_{x=0}$$

A:

mandatory

Note: If you do not know the answer to this question, reload the page and you'll get another question.

Cuprins

- Filtre de text
  - grep, cut, tr, sed, awk etc.
- Programare în shell
  - if, for, seq, while, read, funcții
- Exemple de scripturi

Suport curs

- Suport (Introducere în sisteme de operare)
  - Capitolul 12 – Shell scripting
    - Secțiunile 12.4, 12.5, 12.6, 12.9

Remember, remember ... cursul 6

- Inlantuirea comenzilor
- Operatori shell, caractere speciale
- Escaping/citare
- Variabile shell
- Variabile de mediu
- Globbing
- Expresii regulate

Filtre de text

- All is text. (Unix)
- Comenzi care “filtrează” text
  - `filtru < input_file > output_file`
  - `command1 | filtru | command2`
- În Windows PowerShell se lucrează cu obiecte
- Operații
  - selectare linii
  - selectare coloane (după separator)
  - prelucrare linii (modificare elemente)
  - contabilizare

cat, tac, rev, nl

- cat
  - afișează intrarea standard la ieșirea standard
  - Ce efect are rularea comenzii cat fără nici un argument?
    - CTRL+D
  - **`cat -n`**
- tac
  - afișare inversată (prima linie este afișată ultima)
- rev
  - afișare inversată a fiecărei linii (primul caracter este afișat ultimul)
- nl – number lines

head, tail

- **`head -10, head, head -n 10`**
  - primele 10 linii
- **`head -n -20`**
  - toate liniile mai puțin ultimele 20
- **`tail -10, tail, tail -n -10`**
  - ultimele 10 linii
- **`tail -n +21`**
  - toate liniile începând de la linia 21 (adică mai puțin primele 20)

cut

- Selectare coloane de text
- Separatorul implicit este TAB
- `cut -f 1 < file`
- `cut -f 1-3 < file`
- `cut -f 1,4 -d ':' < /etc/passwd`

tr

- Transliterate
- Transformare litere în alte litere
  - `tr 'a' 'b' < file`
  - `tr 'a-z' 'A-Z' < file`
- Ștergere
  - `tr -d ' ' < file`
- Squeeze
  - `ifconfig | tr -s ' ' | cut -d ' ' -f 2`

– `tr -d -c 'A-Za-z' < /dev/urandom | head -c 8`

sort, uniq

- Sortarea liniilor
- Implicit alfabetică
- **`sort -n file`** – sortare numerică
- **`sort -r file`** – reverse sort
- **`sort -u file`**, **`sort file | uniq`**
- **`sort -k 3 file`** – sortează în funcție de coloana 3

wc

- Contorizarea elementelor
- Cu argument fișier afișează **și numele fișierului**
- **`wc -l`** – numărul de linii
- **`wc -c`** – numărul de caractere
- **`wc -w`** – numărul de cuvinte

grep

- Extrage linii ce conțin un anumit pattern
- Folosește expresii regulate
- **`grep 'pattern' file`**
- **`grep -r 'pattern' /path/to/folder/`** - căutare recursivă
- **`grep -v 'pattern' file`** – invert match
- **`grep -i 'pattern' file`** – ignore case
- **`grep --color=auto`**
  - recomandăm folosirea unui alias

Programare în shell

- Variabile (vezi cursul 6)
- Instrucțiuni de decizie
- Instrucțiuni de ciclare
- Funcții

if

```
if grep 'dan' /etc/passwd; then
   echo "OK"
else
   echo "NOK"
fi

if test -f /path/to/file; then
   echo "File exists"
fi

if test $# -ne 1; then
   echo "Usage: $0 file"
fi
```

if (2)

- Primește ca argument o comandă
- 0 = succes -> condiția este îndeplinită
- !0 = insucces -> condiția nu este îndeplinită
- În general comanda test
  - `test -f file`
  - `test -d file`
  - `test string1 = string2`
  - `test num1 -eq num2`
  - `test -z string1`

for, seq

```
for i in 1 2 3 4 5 6 7 8 9 10; do … done

for ((i = 1; i <= 10; i++)); do … done

for i in $(seq 1 10); do … done

for i in $(seq -f "%02g" 1 10); do … done

for f in *; do … done

for user in $(cut -d ':' -f 1 < /etc/passwd); do … done

for arg in $@; do … done
```

while, read

```
while read a b c; do … done < file

IFS=':'; while read uname pass uid extra; do
   if test $uid -ge 1000; then
       echo "$uname"
   fi
done < /etc/passwd
```

while, read (2)

- Primește ca argument o comandă (la fel ca if)
  - de obicei read
- read
  - întoarce 0 când a citit o linie
  - întoarce diferit de zero la sfârșitul fișierului (EOF)
  - folosit
    - stand-alone (similar fgets, scanf)
    - împreună cu while
- IFS
  - input field separator
  - specifică argumentul care va fi folosit de read

Funcții în shell

```
hello()
{
    echo “hello”
}

ip_for_hostname()
{
    host “$1” | head -1 | cut -d ` ` -f 4
}

DEBUG()
{
    test “$_DEBUG” -eq 1 && $@
}
```

Funcții în shell (2)

- Modularizare
- func_name() { … }
- Se apelează ca o comandă (func_name)
- Poate folosi return
  - dacă nu, valoarea de retur a funcției este valoarea de retur a ultimei comenzi rulate
- Argumentele transmise sunt prelucrate cu ajutorul variabilelor specifice scripturilor shell
  - $#, $1, $2, $@

sed

- **`sed –n ‘/abc/p’`**
  - echivalent grep
- **`sed –n ‘$=‘`**
  - echivalent wc –l
- **`sed ‘10q’`**
  - echivalent head
- **Substitute**
  - **`sed ‘s/ana/bogdan’`**
  - **`sed ‘s/ana/bogdan/g’`**

sed (2)

- Stream editor
- Citește linie cu linie și face prelucrări
- Sintaxă spartană
  - range + comenzi
  - **`$=`** (la sfârșitul fișierului, afișează numărul liniei)
  - **`10q`** (la linia 10, quit)
  - **`s/ana/bogdan/g`** (peste tot, substituie ana cu bogdan)
  - **`1,10s/ana/bogdan/g`** (în primele 10 linii, substituie)
  - **`/corina/s/ana/bogdan/g`** (pe liniile ce conțin corina, substituie)
- Sintaxa de substituție este folosită și în vi

awk

```
ifconfig | awk -F `[ \t]+' `{print $5;}'

dpkg -l `apache2*' | grep ^ii | awk -F `[ \t]+' `{print
  $2;}'

< /etc/passwd awk -F `:' `{ if ($3 >= 1000) print $1;}'
```

awk (2)

- cut++
  - permite specificarea unei expresii regulate ca separator
- Limbaj de programare – sintaxă similară C
- Referă câmpurile prin variabile interne
  - $0 – întreaga linie
  - $1, $2, …
  - NR – number of records (număr de linii)
  - NF – number of fields (număr de coloane)

Recomandări shell scripting

- Nu folositi shell scripting pentru ceea ce se face mai bine/usor/eficient in Python, C, Java, Ruby, PHP etc.
- Modularitate (do one thing, do one thing well)
- Nu reinventați roata
- Cleanup actions
- În general, nu folosiți majuscule în numele variabilelor
- Nu parsați ieșirea comenzii ls (folosiți stat, for i in *, find sau arrays)
- Folosiți ghilimele când referiți variabile
- http://mywiki.wooledge.org/BashFAQ

Upload MAC address

```
#!/bin/bash

declare IFACE=eth1
declare URL="http://anaconda.cs.pub.ro/~razvan/school/uso/lab/macs/putmac.php"
declare MY_HWADDR

function getmac()
{
    MY_HWADDR=$(/sbin/ifconfig "$IFACE" | grep HW | awk -F '[ \t]+' '{print $5}')
}

function uploadmac()
{
    wget -o /dev/null "$URL?mac=$MY_HWADDR" &> /dev/null
}

function main()
{
    getmac
    uploadmac
}

main

exit 0
```

Run torrent script

```
#!/bin/bash

BTDOWNLOADHEADLESS=/usr/bin/btdownloadheadless

TORRENT_FILE="$1"
DOWNLOAD_DIR="$2"

function check_alive()
{
   ps -ef | grep $BTDOWNLOADHEADLESS | grep $TORRENT_FILE &> /dev/null
}

check_alive
if test $? -ne 0; then
   pushd . &> /dev/null
   cd $DOWNLOAD_DIR && nohup $BTDOWNLOADHEADLESS $TORRENT_FILE &>
   /dev/null &
   popd &> /dev/null
fi

exit 0
```

Mass add users

```
#!/bin/bash

IFS=","
INPUT_FILE=lists/cdl_2010_list.txt

while read username email name surname; do
   ./ldap_add $username $email $name $surname
done < $INPUT_FILE

exit 0
```

Migrare utilizatori

```
#!/bin/bash

#passwd_file   fisier cu useri gen /etc/passwd

while read user skel_pass uid gid comm home shell; do
   save_dir="/home/$user/old_anaconda"

   echo "$user $comm $home $shell"

   useradd -m -d /home/$user -g users -s $shell -c "${comm}" $user

   mkdir ${save_dir}
   chown $user:users ${save_dir}
   bash ./rsync.sh $home ${save_dir}

done < passwd_file
```

Send event

```
#!/bin/bash

# get subject
if test $# -ge 1; then
    subject="$1"
else
    echo -en "Subject: "
    read subject
fi

# get file
if test $# -ge 2; then
    file="$2"
else
    echo -en "File: "
    read file
fi

# send mail to all LDAP accounts
for i in $(ldapsearch -x mail | grep mail: | cut -d " " -f 2); do
    mail -s "$subject" $i < $file
done

exit 0
```

USO just crunch it

Wanna see something real scary?

```
#!/bin/sed -f

{
s/\[\(http[^]]\+\)]/[[\1]]/g
s/<pre>/<code>/g
s/<\/pre>/<\/code>/g
s/<source[ \t]\+lang="\([a-z]\+\)">/<code \1>/g
s/<\/source>/<\/code>/g
s/<tt>/{{{/g
s/<\/tt>/}}}/g
s/<b>/**/g
s/<\/b>/**/g
s/<ul>//g
s/<\/ul>//g
s/[ \t]*<li>\(.*\)<\/li>/* \1/g
s/'''/**/g
s/''/\/\//g
s/{{{\[\[/[[/g
s/\]\]}}}/]]/g
s/\(\[\[http:[^ ]\+\) /\1|/g
}
```

Cuvinte cheie

- filtre de text
- cat
- tac
- nl
- head
- tail
- sort
- uniq
- cut
- tr
- grep
- if
- case
- for
- while
- read
- IFS
- funcții
- sed
- awk

Link-uri utile

- http://www.shelldorado.com/
- http://tldp.org/LDP/abs/html/
- http://linuxcommand.org/writing_shell_scripts.php
- http://sed.sourceforge.net/sed1line.txt
- http://www.gnu.org/manual/gawk/gawk.html
- http://mywiki.wooledge.org/BashFAQ

The End